Development and Evaluation of Computer-Aided Assessment in Discrete and Decision Mathematics

A Thesis Submitted for the Degree of Doctor of Philosophy

by

Kinga Zaczek

School of Information Systems, Computing and Mathematics

Brunel University

September 2014

ABSTRACT

This thesis describes the development of Computer-Aided Assessment questions for elementary discrete and decision mathematics at the school/university interface, stressing the pedagogy behind the questions' design and the development of methodology for assessing their efficacy in improving students' engagement and perceptions, as well as on their exams results. The questions give instant and detailed feedback and hence are valuable as diagnostic, formative or summative tools.

A total of 275 questions were designed and coded for five topics, numbers, sets, logic, linear programming and graph theory, commonly taught to students of mathematics, computer science, engineering and management. Pedagogy and programming problems with authoring questions were resolved and are discussed in specific topic contexts and beyond. The delivery of robust and valid objective questions, even within the constraints of CAA, is therefore feasible. Different question types and rich feedback comprising text, equations and diagrams that allow random parameters to produce millions of realisations at run time, can give CAA an important role in teaching mathematics at this level.

Questionnaires identified that CAA was generally popular with students, with the vast majority seeing CAA not only as assessment but also as a learning resource. To test the impact of CAA on students' learning, an analysis of the exam scripts quantified its effect on class means and standard deviations. This also identified common student errors, which fed into the question design and editing processes by providing evidence-based mal-rules. Four easily-identified indicators (correctly-written remainders, conversion of binary/octal/hexadecimal numbers, use of correct set notation {…} and consistent layout of truth tables) were examined in student exam scripts to find out if the CAA helps students to improve examination answers. The CAA answer files also provided the questions' facilities and discriminations, potentially giving teachers specific information on which to base and develop their teaching and assessment strategies.

We conclude that CAA is a successful tool for the formative/summative assessment of mathematics at this level and has a positive effect on students' learning.

Acknowledgements

First and foremost I want to thank my supervisor Dr Martin Greenhow for supporting me during this PhD. His guidance, time and patience with me are much appreciated. He always believed in me and motivated in tough moments. His enthusiasm, smile, good sense of humour and understanding allowed me to complete this research.

I want to thank Dr Steven Noble, my second supervisor, for his support with the research, administration staff for friendly atmosphere and finding the necessary data, computer support officers for help anytime something was not working, all the lecturers and fellow PhD students for good word and rich conversations.

I also thank my friends and groups I was a part of. I met fantastic people on squash courts, during Polish Society meetings, but also when living in the hall. We had fun times but also very supportive conversations if needed. Friends outside the university gave me not any less support. They motivated me, believed in me and were always for me when I needed them.

Last, but not least, I thank my family for all their love and encouragement. For my parents who raised me as a person who respects values. They gave me the opportunity of studying in the UK hoping it will give me a better life in future. They, my sister and my beloved grandparents made me feel loved and special. I am happy they feel proud of me. I would not make it this far without you. Thank you.

Contents

The Appendices (A-E) are included as separate files.

List of Figures

List of Tables

1. Introduction

The objective of this study is to develop and evaluate Computer-Aided Assessment (CAA) for topics in discrete and decision mathematics commonly taught to students of mathematics, computer science and certain branches of engineering and management. As a result, a set of questions were developed and different types of data were collected and analysed. Anonymous questionnaires were completed by students in order to give them an opportunity to express their thoughts about the CAA and its implementation. Another source of information was the answer files from the CAA. They provide hard evidence on students' engagement with tests, facility and discrimination of questions students are exposed to in order to improve their learning. This leads to steps that should be taken by teachers in lessons or to solve any issues with coded questions. Finally exam scripts and final examination results were examined to ascertain the possible impact of CAA on students' learning.

This chapter introduces the concept of assessment and describes its different types. CAA is widely used and therefore an outline of other available software will be presented. This chapter describes the technology used, introduces the research questions that underpin this study and explains the methodology that was used to answer them. In the final section, the structure of this thesis is presented.

1.1. Types of assessments

The main types of assessment (see Bull & McKenna, 2004) can be categorised as:

- Diagnostic – such assessments are usually given at the start of a course to ascertain the student's current knowledge and skills base,
- Formative – such assessments aim to reinforce and augment students' learning by (usually regular) testing and feedback,
- Summative – such assessments are used to grade students, often by end-of-term examinations. In mathematics the marking is usually criterion referenced (i.e. do they know a specified body of mathematics) and seldom norm referenced (i.e. how well does a student compare with his/her peers). Having said that, it is common practice to scale marks in some modules and there can be defensible reasons for doing so,
- Mastery – such assessments usually involve reaching a (usually very high) mark on basic material, without which further progression is impossible. For example,

student may be required to get 80% on a basic arithmetic and algebra test before being allowed to progress in a first-year calculus module.

Of course, any assessment may contain elements of all of the above; for example, an initial class test in the first year would be diagnostic (especially to see which students need additional support), formative with model solutions being discussed in a subsequent tutorial and summative, usually counting a small percentage towards the modules overall grade (this can be as low as 2% and still assessment-driven students will take the assessment seriously).

The main focus of this thesis pertains to the border between formative and summative assessment. The questions aim to deliver a learning resource in its own right (especially by writing very extensive feedback screens) and to build student confidence. Acknowledging the fact that some, usually weaker, students will only do assessments for marks, we have allocated 20% of the module mark to the five assessments described here (4% each), but with additional safeguards against cheating in various ways (aliasing or using illegal software are the main ones) being implemented. This does not mean that the questions cannot be used for other assessment types and there is no restriction at the question level on their use. Certainly one then needs to consider the fairness (as opposed to the validity) of the marking and the dichotomous marking scheme used here may need to be altered to fit the test's purpose. Such a question lies beyond the scope of this thesis, but it is notable that no student has ever queried the fairness of dichotomous marking (perhaps they are unaware of its use when doing the tests). In other systems, such as Dewis (see below), questions do not allocate marks but simply pass flag values to other software to be post-processed according to any marking scheme that reflects the purpose of the test; this is probably the correct way of doing the marking, but we cannot implement this within QM Perception that underlies the present Mathletics system.

For computer-aided assessment it is important to split the questions into:

- Subjective
- Objective

Only objective questions, that do not require value-judgements on behalf of the marker, can be delivered by any CAA system at present, including the present one. Typically a written exam would contain both types of question, objective ones focussing on 'getting an answer' and subjective ones on supplying a proof or giving an explanation or justification of a result. The first would correspond to the lower-levels of Bloom's taxonomy (Forehand, 2014) and can be regarded as *necessary but not sufficient* to assess students' understanding. Conversely, the second is much more synoptic in its scope and could only be tackled successfully by students with the content of the objective questions firmly in place: this corresponds to Bloom's high-levels of assessment. Questions can be tagged according to Bloom's taxonomy, but the consensus between e-assessment authors seems to be that it is not a useful thing to do. This may be partly because the taxonomy uses words that mathematicians find ambiguous or confusing. For example, Bloom's second-highest level is 'evaluate' which means 'form an opinion or judgement on' not 'find the value of' as in "Evaluate the following integral" which is the mid-level skill of 'applying' known techniques. The CAA questions delivered here can be categorised as: 'remembering' e.g. factual recall of a definition, 'understanding' e.g. knowing what is required by the question and applying a correct procedure (not "understanding some mathematics" as a mathematician would assume which is a much higher-level skill) and 'applying', see above. Above this, 'analysing' and 'evaluating' would generally be assessed in an examination using at least partly-subjective questions (e.g. write a proof or draw a conclusion from some data) whilst 'creating' would more naturally fall into project work or postgraduate study.

1.2. CAA in mathematics in Higher Education

An e-Assessment in Maths Working Group lead by Foster (2013) has produced an overview of e-assessments developed in the UK that offer open-access:

- *maths e.g.* http://www.mathcentre.ac.uk:8081/mathseg/
- DEWIS http://www.cems.uwe.ac.uk/dewis/welcome/index.html
- STACK http://stack.bham.ac.uk/moodle/
- Numbas https://numbas.mathcentre.ac.uk/
- QTIWorks https://webapps.ph.ed.ac.uk/qtiworks/

Comparison and technical specification of the above is given in e-Assessment in Maths (2013).

In addition the most popular commercial systems in use in the UK today appear to be:

- Maple T.A. http://www.maplesoft.com/products/mapleta/
- MyMathLab http://www.pearsonmylabandmastering.com/northamerica/mymathlab/
- Wiley Plus https://www.wileyplus.com/WileyCDA/

It is beyond the scope of this thesis to compare these disparate systems, still less recommend one, which can hardly be done without knowing the intended use (high or low stakes, formative or summative) and target group (mathematics students or others that have significant mathematical content within their courses, basic or more advanced topics). Each system has its pros and cons and a potential user would need to investigate the above web sites in detail and also to specify exactly what is required from the use of CAA and how it would be embedded. This often requires internal approval in universities, often with a substantial lead time for approval, implementation and testing; introducing CAA is seldom feasible with a current academic year.

1.3. Technology used

The main philosophy behind the question creation process was to maintain complete control of what is delivered to students, both in terms of question content and question functionality. Thus many existing systems, especially those that come as part of a VLE, are quite inadequate for the delivery of even rather simple mathematical content. Consequently we have used Questionmark Perception version 3 (Questionmark Perception, 2014). This is a commercial system and provides a test-delivery and analysis shell. We use version 3 since later versions prohibit authoring of questions at the level of control required to allow effective use of random parameters. It is worth noting that all of the questions are in fact web pages and can be delivered via the *maths e.g.* interfaces (Maths E.G., n.d.) that is independent of any third-party software and freely delivered via a Creative Commons licence.

Within the Perception version 3 shell a question author can either use standard wizards (inadequate for mathematics) or script one's own content using the following:

- JavaScript (MDM, 2014) is a standard and powerful scripting software for web pages. It includes ordinary html.

- MathML (W3C, 2014) is a standard and powerful mark-up language for the display of mathematics on web pages. It can be delivered in a number of ways: the present work uses the WebEQ applet (Design Science, 2014), but, since applets do not work natively on modern mobile devices (phones and I-pads) future work should re-purpose the equations for use with MathJax (MathJax, 2014).

- SVG (Scalable Vector Graphics, 2014) is a powerful mark-up language for the display of diagrams on web pages. It is now native to most browsers and hence no longer requires a plug-in.

It is important to realise that no static graphics are used: all equations use MathML and all diagrams use SVG; both are generated at runtime, thereby responding to the values of the question's random parameters e.g. graphs are created dynamically so that the number of vertices and edges change with each realisation. A further benefit is that the student's display preferences (font size, colour and background colour) is inherited by the equations and diagrams which is known to be of benefit to some dyslexic and partially-sighted students (British Dyslexia Association, n.d.).

1.4. Research questions

Six main research questions for this study were specified from the outset or arose during the course of the study. These are:

1. How might CAA influence teaching and learning?
2. Is it feasible to write questions in the areas of discrete and decision mathematics?
3. What are the most difficult and most discriminating questions within a particular topic?
4. How does the formative feedback affect the results students achieve in exams within a particular topic?
5. Can we establish the methodology on how effective the questions are?
6. Can we apply the techniques of CAA developed here more widely?

These questions will be answered throughout the Chapters 3 – 10 and addressed in the concluding Chapter 11.

1.5. Methodology

Hemmings and Hollows (2005) claim that it is very important for the research to concentrate around the research questions rather than only the topic. These then help specify an appropriate methodology. The methodology of this study can be represented graphically as in Figure 1.5.

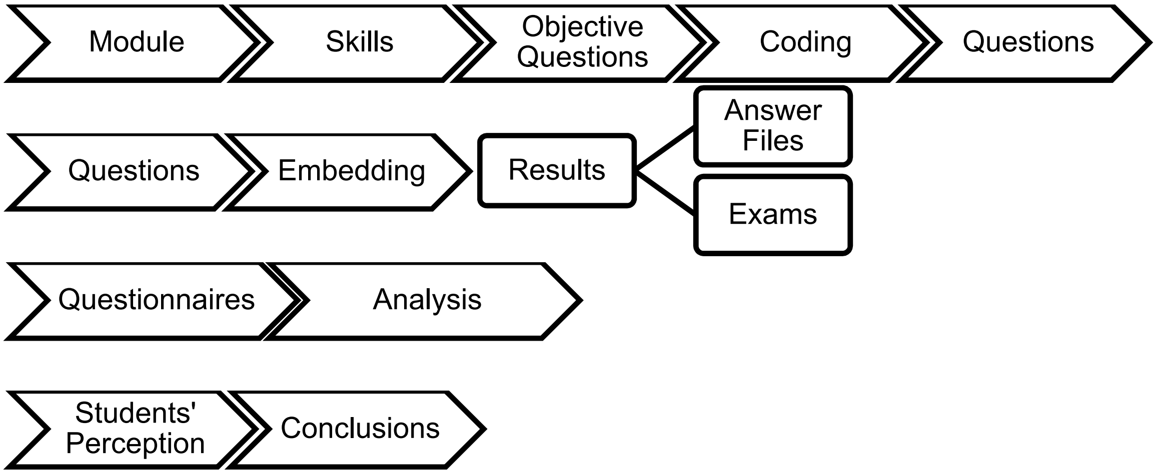


Figure 1.5: The structure of the research and its methodology

The top line of Figure 1.5 refers to the question creation process and addresses research question 2. It describes staff-related activities. The first two items in the second line are also staff-related and address research questions 1 and 6, whilst the second two are student-related and address research questions 3, 4 and 5. The items in line 3 are again student related and address research question 4. Finally the staff-related activities of analysis of this data and drawing conclusions address all research questions except 2.

The above rather general research questions are answered within the context of CAA for Discrete and Decision Mathematics taught to a large cohort of foundation-year students i.e. at the school/university interface. For each topic involved in the module a number of skills has to be specified. This is a rich list of assumed skills (prior competencies) and tested skills (competencies gained in the process of study each topic). Their specification is important in the overall teaching process in order to provide students with the necessary information at the correct level. For CAA it is even more important since it is crucial to specify all such skills when formulating objective questions, especially those to be delivered by a CAA system. It is important that formulated questions target exact skills for testing. Since well-formulated objective questions have unambiguous answers they are well suited to CAA and automatic marking (Bull & McKenna, 2004). Moreover, objective questions can be made clear in their presentation and have full feedback, thereby fulfilling an important formative role. Following these steps a large number of questions was created as described in Chapters 3 – 7 giving answers to the research questions: 'Is it feasible to write questions in the areas of discrete and decision mathematics?' and 'Can we apply the techniques of CAA developed here more widely?'

After the questions were coded and tested they were embedded for wider usage. The way the test can be used depends on the teacher. We chose to take the best of the first five attempts at any test. These tests generate hard data in the answer files from the CAA which were analysed to give the questions' facilities and discriminations, answering research question 3: 'What are the most difficult and most discriminating questions within a particular topic?' This can be used to inform teaching by indicating topics on which more/less time can be spent and to help to set effective assessments, addressing research question 1: 'How might CAA influence teaching and learning?'. Moreover, teachers can then have a picture of students' engagement with topics and may prompt changes in the examination or in the order in which topics are taught if this affects the preference at the exams or CAA. Exam scripts were analysed with respect to four indicators which give an indication of certain mathematical skills. They were possible to identify easily in both the CAA and in the exams. They were measured across years when there was no CAA and after it was introduced. Students' engagement together with the exam results will help to answer research question 4: 'How does the formative feedback affect the results students achieve in exams within a particular topic?'.

Another source of information comes from the questionnaires that were distributed among students. They were used to assess students' perception about the CAA and how they perceive it. This evidence together with the data on indicators, facility, and discrimination will be used to answer research question 5: 'Can we establish the methodology on how effective the questions are?'

1.6. Structure of the thesis

This thesis mostly reflects the order in which the research was conducted. Questions spanning five topics (numbers, sets, logic, linear programming, and graph theory) were constructed: together this forms the principal contribution of this research. Chapter 2 outlines information about target group, question database and questions themselves. Each topic is described separately in chapters 3 to 7, each organised in the same way. Subtopics (e.g. for numbers these are prime factorisation, modular arithmetic, non-decimal arithmetic, scientific notation) form subchapters, which discuss the assumed and tested skills, the characteristic of the objective questions (i.e. question types and styles, randomisation), feedback and the technical content.

This technical content describing the design of the questions is followed by the evaluation of CAA consisting of three chapters. Chapter 8 discusses students' perceptions of the tests and looks at whether responses to different questionnaire questions depend on each other. Chapter 9 presents the hard data from CAA. The answer files allow the unprejudiced analysis of CAA on students' engagement, as well as giving the facility and discrimination of each question. Chapter 10 analyses exam performance throughout seven years' worth of data, firstly with no access to CAA and then with it being widely used by students. The exam scripts have also been used to determine most common mistakes later used as distractors in multiple choice questions or to match feedback to submitted answers. Chapter 10 investigates the relations between the CAA results and the exam results.

The thesis finishes with conclusions and recommendations for further study.

The CD contains the electronic version of this thesis as well as all the appendices that include the question coding (QML files) and spreadsheets of raw data. This is included to facilitate further question development and analysis of the data to address other research questions, perhaps along the lines suggested in the Recommendations.

2. Discrete and Decision Mathematics questions

The Discrete and Decision Mathematics module used as a vehicle for this study consists of five topics: Numbers, Sets, Logic, Linear Programming and Graph Theory, taught in this order. This chapter includes information about target group, question database and general appearance of questions, question types used across the topics, randomisation of parameters and feedback.

2.1. Target group

The target group of this study is a cohort of roughly 100 Foundations of IT (FoIT) students taking the compulsory Discrete and Decision Mathematics module at Brunel University during the years 2005/6 to 2011/12. This group of students usually lacks mathematics A-levels at grade C or above (typically only about 30% of cohort has this or equivalent).

It is assumed that students' knowledge of mathematics is at GCSE level at grade C or better. From the content of the syllabus and the specification for Edexcel GCSE in Mathematics (2012), students are expected to know for example negative numbers, times tables, factors, multiples, fractions, straight-line graphs.

2.2. Question database and general appearance

The database on Discrete and Decision Mathematics consists of 275 questions (Appendix E contains the QML files with all the questions). The database is separated into five topics, as specified at the start of this chapter. Questions on logic and sets can be categorised as Boolean algebra, while linear programming is a topic in decision mathematics. Each of the five topics is split into subtopics which in some cases are split further to indicate, for example, special types of truth tables or algorithms for minimum spanning trees. Such structure in the database allows easy access to questions where one typically wants to test students on specific topics and not on others. Figure 2.2a shows how this database is organised in Question Manager.

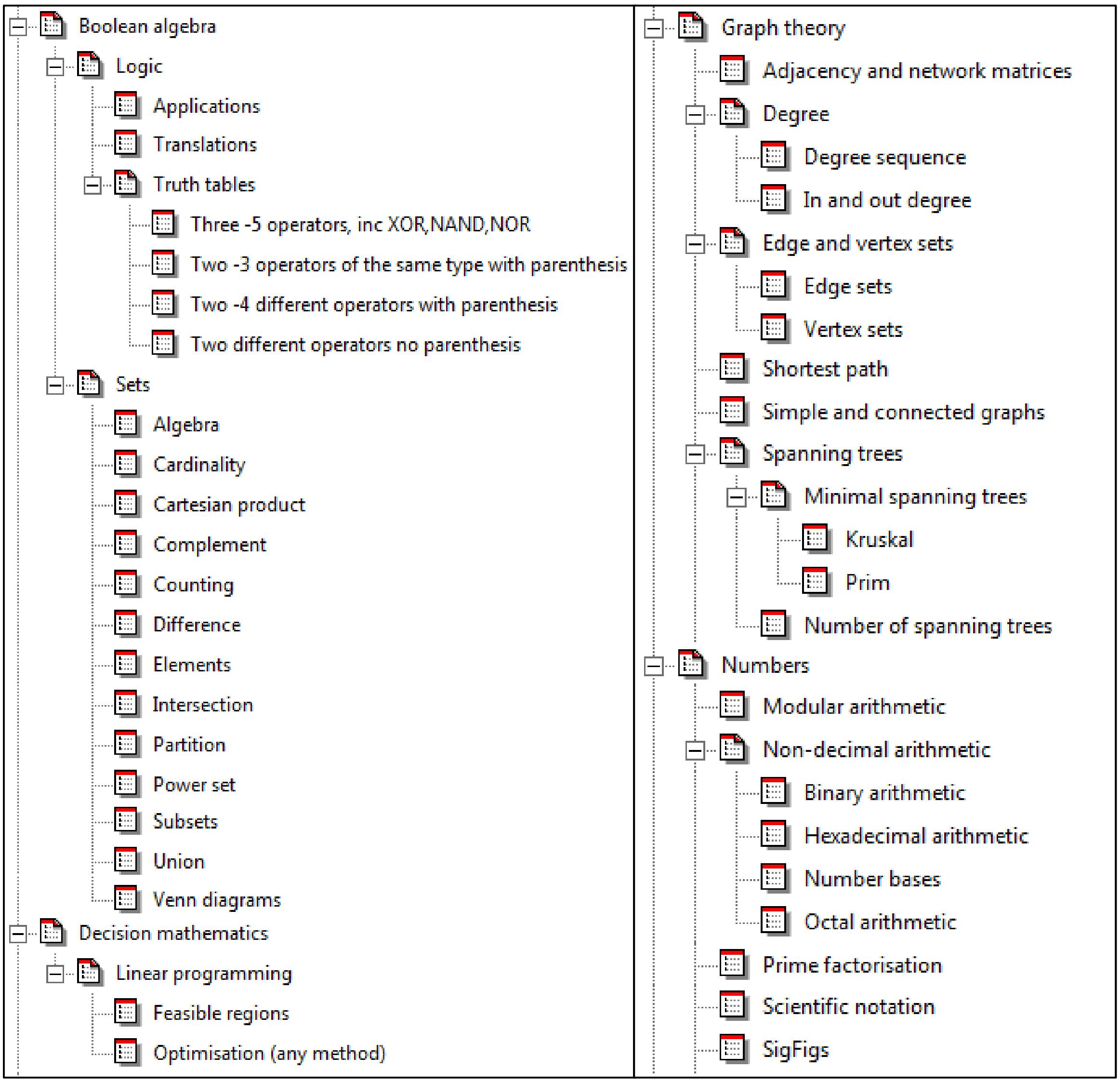


Figure 2.2a: Screen shot of the database on Discrete and Decision Mathematics.

After selecting the subtopic in which questions are located, the full list of questions displays. There is an option to organise columns with respect to how much information one wants to see about questions. As in Figure 2.2b, these can include the question description, the date of last modification, status (e.g. normal if questions are completed, incomplete if they are under development, retired if they are not to be currently used due to possible newer version of the question being available) and, importantly, the template (JavaScript coding of functions written for specific topics accessed by most or all questions within that topic).

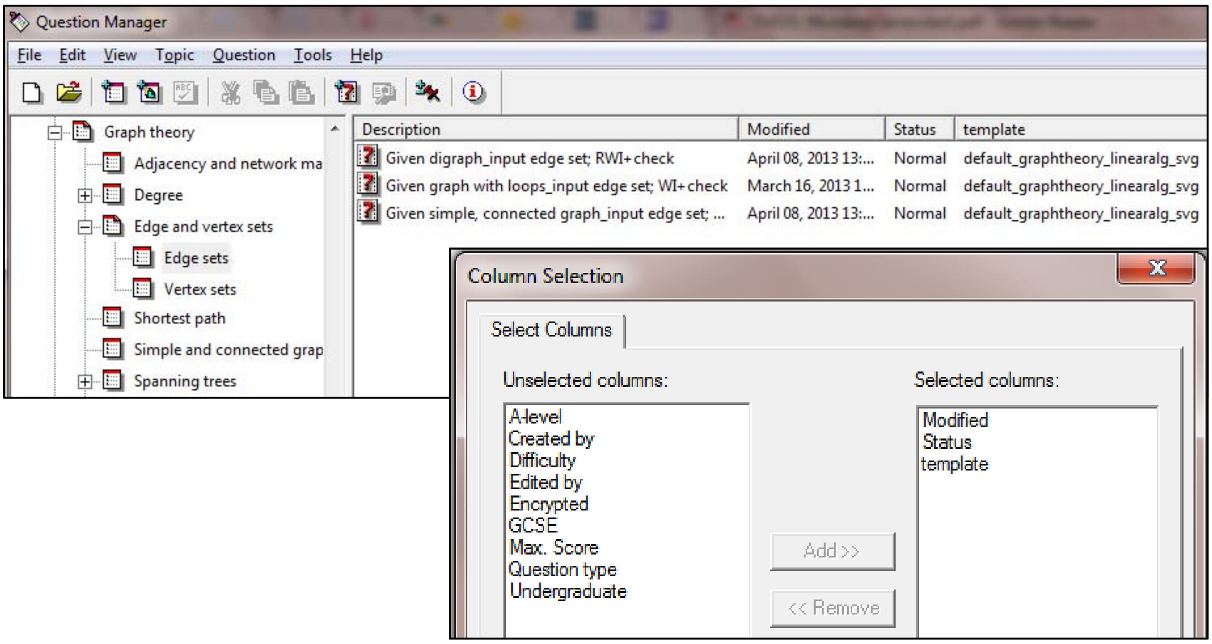


Figure 2.2b: Screen shot of the information available for a selected subtopic, i.e. Edge sets.

Another feature of Question Manager is a possibility to assign tags to describe/classify questions. It is possible to provide the information on the level of mathematics and the difficulty. The database under this research spans mathematics GCSE, A-level D1 and Undergraduate-Foundation levels. The questions can have easy, intermediate, hard difficulty level. Templates necessary to run the questions should be indicated here as well. Figure 2.2c shows an example of all three tags being assigned to one question on linear programming, optimisation. This enables searching through the database according to the tags when creating assessments and this is a notable feature of the *maths e.g.* teacher interface given that teachers will know what A level module they are teaching but are unlikely to have the time to search through the extensive database of over 2000 questions to find what they want.

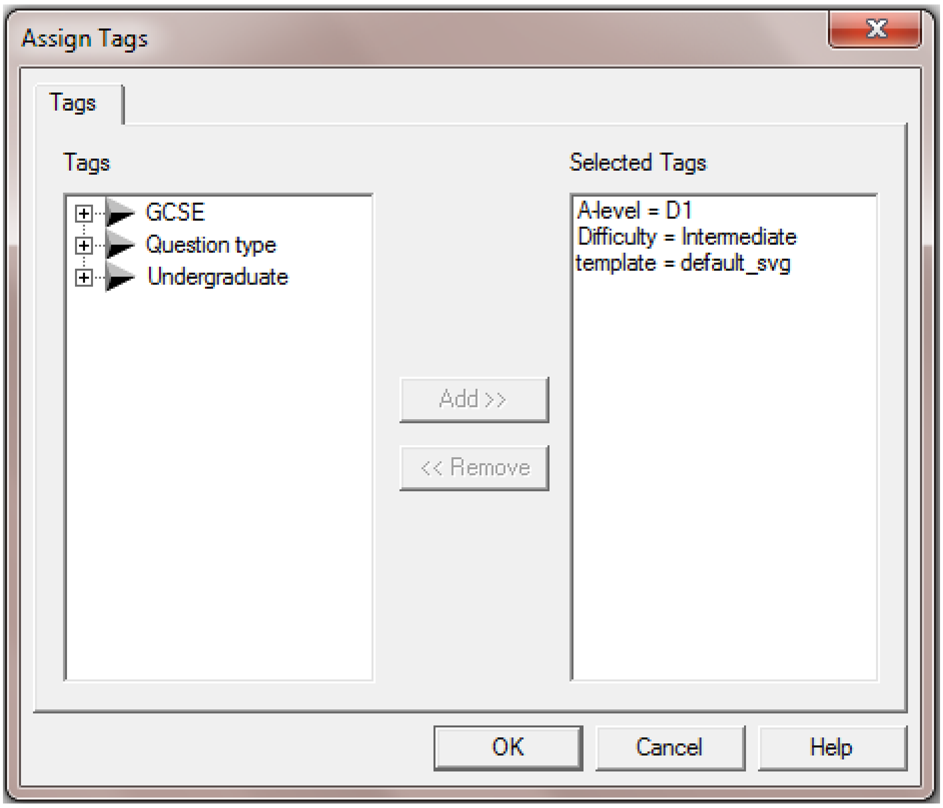


Figure 2.2c: Screen shot of the 'Assign Tags' dialog box.

A feature of Mathletics and *maths e.g.* is the possibility to change colours and fonts of displayed questions. A dialog box is available in the top right corner of each question. It allows change of the font size, colour and type, as well as colour of the background (Figure 2.2d is the screen shot of the question on degree sequence showing the enabling changing fonts and colours, while Figure 2.2e shows the feedback screen for the same question with changed colours and font). This is to support students with learning difficulties, i.e. visual stress caused by dyslexia or partial sight. According to the British Dyslexia Association (n.d.) about 35-40% of pupils with dyslexia have problems when reading from the white background, therefore possibility of changing the colours helps to overcome that problem.

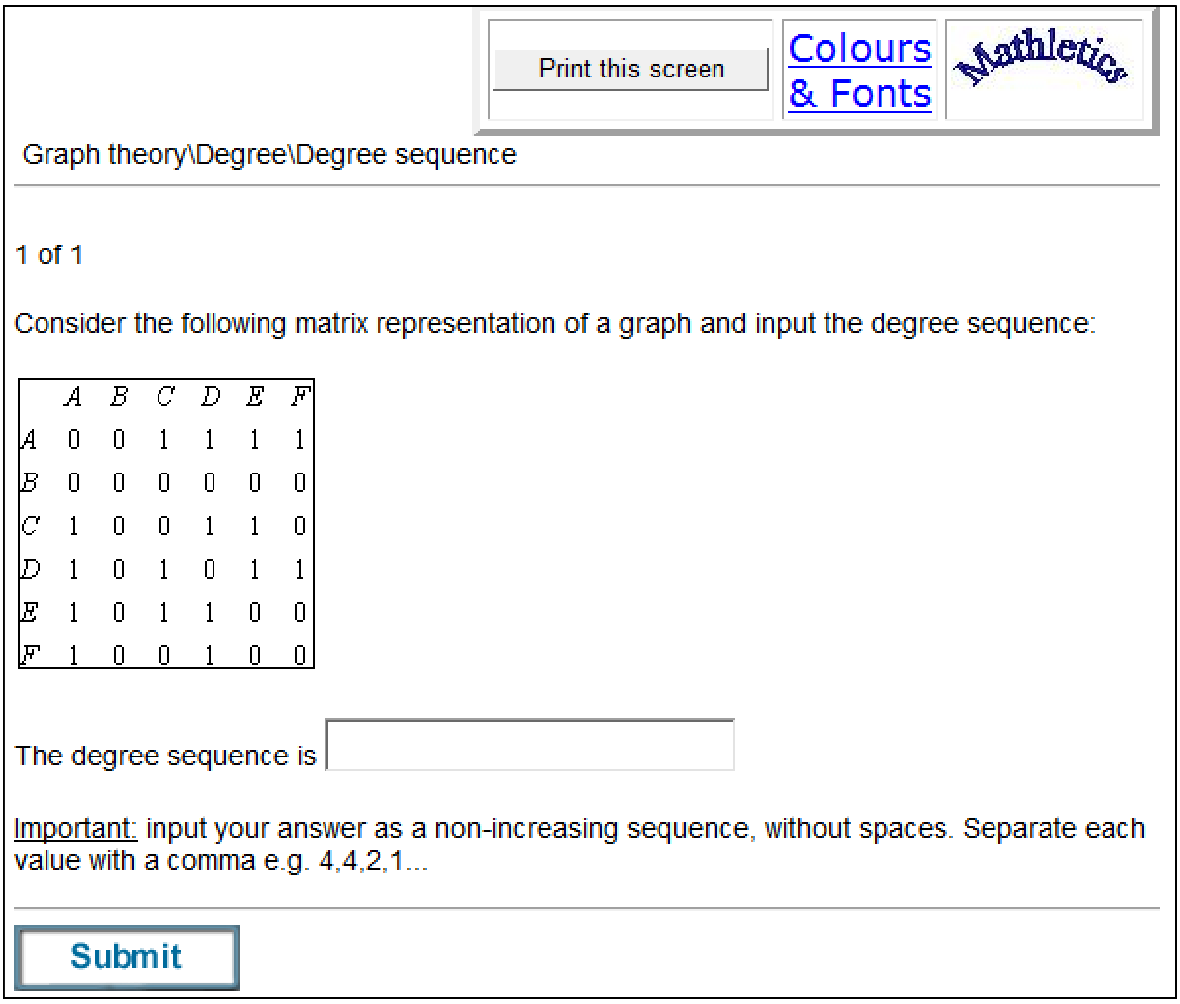


Figure 2.2d Screen shot of question on degree sequence showing fonts and colours link.

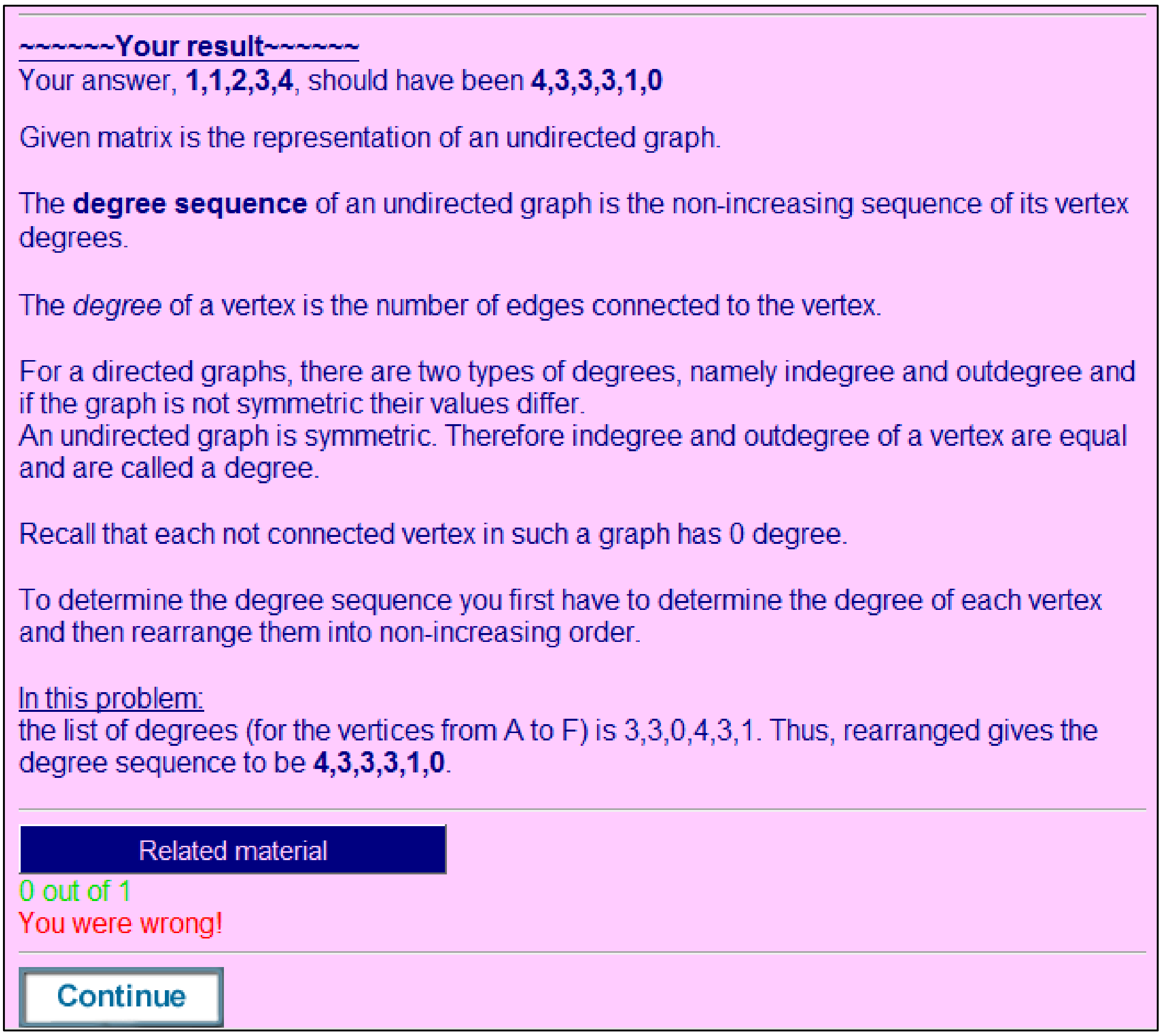


Figure 2.2e: Screen shot of the feedback screen (without question text that also displays but is included in Figure 2.2d) showing the changes made to fonts and colours.

In addition to the full feedback, students are provided with a possibility to access additional resources (references to textbooks and/or webpages, a template containing the relevant webpages can be found in Appendix E) through the 'Related material' button (see Figure 2.2e).

2.3. Question types

The database comprises questions of different types. They add variety to tests but also facilitate valid assessment of tested skills. Some questions require students to select the answer, some require numerical or word input. A brief description, advantages and disadvantages of question types used in this study will be presented below in the mathematics-specific context of the present topic areas. This contrasts with the general advice given in e.g. Burton *et al* (1991) and Chan & Kennedy (2002), which pertains more to textural contexts and is therefore less relevant here. Of more relevance is the very extensive set of multi-choice questions for electronic voting systems given in MathQUEST/MathVote (Carroll College, n.d.) but these do not consider the use of random parameters or feedback.

Multiple choice (MC) questions – participants have to select one of many (six in Mathletics) options (see Figures 5.2.2). Two options that are always available are ‘None of these!’ and ‘I don’t know’. Obviously ‘None of these’ must appear consistently to avoid giving a clue if it were to appear only when it’s correct (generally this is the case 1 in 8 times, at random). The remaining four are randomly selected out of five or more options available, of which one is a correct answer and others are coded distractors which often are based on errors known to be made by students. An advantage of these questions is that they will explain the misconceptions or mistakes that lead students to the submitted incorrect answer. Moreover, since the number of available responses is limited and there is a lot of information on the screen, these questions are generally easier than other types and hence can be used to increase students’ confidence. A disadvantage is that they involve the author in a lot of work to write valid distractors so that none are obviously wrong. Random guessing is possible, therefore one never can be certain whether students knew the answer or were lucky. As a result, these questions cannot be relied on to test a deep understanding of the topic.

Multiple-response (MR2/4, MR3/6) questions (see Figure 5.3) are similar to those of multiple choice. They look very similar but the difference is that in MR questions more than one answer can be selected. In the database two out of four or three out of six questions can be chosen (students do not know how many answers should be selected). The advantages and disadvantages are comparable to those of MC questions, but marking is more of an issue. We allocate positive marks for choosing correct AND for avoiding incorrect options and allocate negative marks for doing the opposite.

Numerical input (NI) questions (see Figures 3.1.2) provide students with a box to input a number as the answer. The advantage of this type of questions is that there is no guidance about answer, as it is in the case of MC or MR questions (Gill, 2007). In order to get the correct answer, students sometimes have to perform a number of steps. If they make even a single mistake the answer is automatically marked as wrong, therefore it can be seen as a disadvantage that no marks for workings are awarded. On the other hand, depending on the purpose of the test, it could be desirable to stress that accuracy and precision is necessary. Two/multiple numerical (2NI/XNI) type questions (see Figure 3.1.1) work the same as those of NI type but more than one input box is provided requiring numerical values. For XNI questions, the number of input boxes is dynamic e.g. one box for each prime factor where there may be a variable number of these. Responsive numerical input (RNI) questions look exactly the same as NI but provide students with richer feedback. The feedback responds to the submitted answer when it was wrong according to mal-rules coded by the question author. It tells participants why they were marked incorrectly (i.e. tells them what the author thinks they did wrong) and provides them with the correct solution (as in every other question types).

Word input (WI) type questions (see Figure 5.1) also include an input box, but here only words or strings of characters or characters and numbers are acceptable. Any response not exactly matching the correct answer string is marked as wrong (marking is done by the string comparison of submitted and correct answers). Therefore, a disadvantage of WI questions is the high chance of not scoring due to misspelling of the answer rather than lacking the knowledge. Similarly to the NI type questions no guidance is given regarding the correct answer. In some questions the answer has to be in a specific format (i.e. no spaces, commas should be used as separators) therefore students are prompted to check their response if it has an incorrect string length. Another option is to pre-process the input answer by removing spaces and/or changing case to lower or upper case (this is often done to the students' inputs and hence is visible to them, e.g. name input questions where for example JAMES is changed to James). These questions are described as WI+check. Similarly, responsive RWI and RWI+check questions combining features of WI and RNI(+check) questions.

True, False, Undecidable (TFU) type questions allow one of these three inputs (see Figure 4.2). Again TFU questions pre-process inputs so that t, f or u is changed to T, F or U and other inputs are rejected before marking. Due to the limited number of possible answers, we expect students to have all (usually four) entries correct in order to have a mark awarded. One can see this as a strict marking scheme, but we consider the questions to be more robust when this is implemented. The option of 'Undecidable' is here thought of in an 'everyday' sense, i.e. there is not enough information given to decide the truth or falsity of a statement, not that it is an undecidable statement in the sense of formal logic.

2.4. Randomisation

All of the CAA systems mentioned in Chapter 1 incorporate random parameters into their questions and we do the same. Thus one actually authors a question algebraically or algorithmically to produce all questions of a class. Sangwin (2013) calls this a question *space*, which certainly encapsulates the algebraic structure of a question. We prefer question *style*, since two questions may share the same space yet be different pedagogically depending on what information is given on the screen (e.g. a formula being given or not, a situation being described in words rather than in mathematics thereby requiring students to do a translation into numbers or algebra etc). Whatever it is called, randomisation is clearly an excellent idea since a typical question can produce thousands of millions of realisations to be delivered to the students (Greenhow, Zaczek and Kamavi, 2011). This makes simply copying impossible and repeated practice worthwhile, at least until the students have 'twigged' that they have really done the question before i.e. they are thinking at a higher, and more desirable, level of abstraction. Over and above this, randomisation can be used for 'surface effect' such as randomised names or places in a question (which students then have to filter out as being superfluous information).

To fully exploit the computer medium the values of the random parameters have to be passed to all components of the question so that correct equations and diagrams are displayed. However, a great deal of care must be taken when choosing their values and this usually requires questions to be *reverse-engineered* from the desired answer format. For example, randomly choosing a 7-digit number to be factorised will produce a question of unknown difficulty, whereas starting with (say) 8 prime factors (possibly repeated) will produce a feasible number for the student to factorise.

For multi-choice and responsive numerical input questions, the choice of random parameter values also affects the underlying mal-rules. For example, if x^2 is required but a mal-rule is chosen and coded to be $2x$, then x must never equal 2. For multi-choice questions, such a choice would yield two options of 4, one marked correct and the other marked wrong, whilst for responsive numerical input questions an input of 4 would be marked correct but possibly awarding marks for incorrect student reasoning. This is less serious, but still not desirable.

Beyond numerical and textual randomisation (Gill, 2007), the scenarios used in questions can change. They do not change the pedagogy of the questions but diminish the possibility that students will answer them by automatically thinking that they have done this question already. Bull and McKenna (2004) refer to two other randomisations within test. The order of the questions can vary between tests and also the selection of the questions themselves. However, this is only possible when there is a possibility of choosing questions from a group of comparable questions. Different styles of questions have to be of comparable difficulty and testing the same skills. This type of randomisation is also used in the setup of the assessment that is part of this study. Each of the five tests consists of five to eight questions, each of which is randomly selected from a set of clone questions in a topic.

These general principles are respected throughout and described in greater detail in the context of the questions themselves in later chapters.

2.5. Feedback

The feedback in Mathletics/*maths e.g.* is automated, full and it has the same structure in every question. It starts with the score achieved so far and repeats the question (not shown in Figure 2.2e for reasons of space). This is followed by the schema, as presented in Figure 2.2e, which starts with the submitted answer and the verification information on its correctness. If the question has not been answered the correct answer is not displayed. This is to encourage students to at least attempt the question, and hence benefit more from the feedback. Of course this does not stop students guessing or simply putting in an arbitrary number, known as ‘chatter’ by Walker, Gwynllyw and Henderson (2015). If the answer is correct students are praised with ‘Well done!’; if wrong the correct answer is given and explained (Shute, 2007). The explanation provides students with any underlying theory, e.g. definitions, formulas, and/or a complete solution to the specific problem given. This full and detailed feedback aims to enhance students’ learning and always uses the random parameters that appeared in the question. If the question requires more than one step work to find the solution, a step-by-step model answer is given, e.g. feedback to the questions on Cartesian product (Figure 4.5.3) and on minimum spanning trees (Figure 7.4.3). In addition to this students may be directed to additional resources where they can find the general information about topic. These linked web resources are more topic specific, rather than focusing on the given question.

The focus on such extensive feedback is largely driven by its immediacy (which contrasts starkly with human marking). If a student has answered incorrectly, he/she wants to know exactly WHY and wants to know NOW because they are engaged with the test (at least at the level of wanting the marks, but usually far more seriously than that). Our experience has been that students often question the feedback in subsequent seminars; in a small minority of cases, the question itself is wrong or ambiguous (and hence corrected) but otherwise the student is incorrect in their thinking and needs further explanation by a tutor (sometimes via email, more often in a class). This follow-up demonstrates an excellent interplay between CAA and regular teaching, although it can get quite heated! An example was a student who input the correct elements of a set, but as a sequence 4,5,6,… not as a set {4,5,6,…} and claimed he had been unfairly treated until the difference (which he did not understand) was explained to him.

3. Numbers

Numbers is the first topic of study taught in the Discrete and Decision Mathematics module at Brunel University's foundation level. Questions from the database used by participants of this module include prime factorisation, modular arithmetic, non-decimal arithmetic, and scientific notation. Each of the subtopics is organised in the same way: assumed and tested skills, objective questions, feedback and technical content.

3.1. Prime factorisation

The prime factorisation of a number is a process of finding all prime numbers whose product is equal to the original number. The topic is accompanied by several theorems, definitions and different solving methods leading to the correct answer. These lead to constructing five questions on prime factorisation.

The questions are of three different types, i.e. word input, multiple word input, numerical input, and of different styles. Three questions ask for the prime factorisation whereas in the other questions it only asks for one of the steps in finding the correct answer.

An example of a question on prime factorisation is presented in Figure 3.1 below.

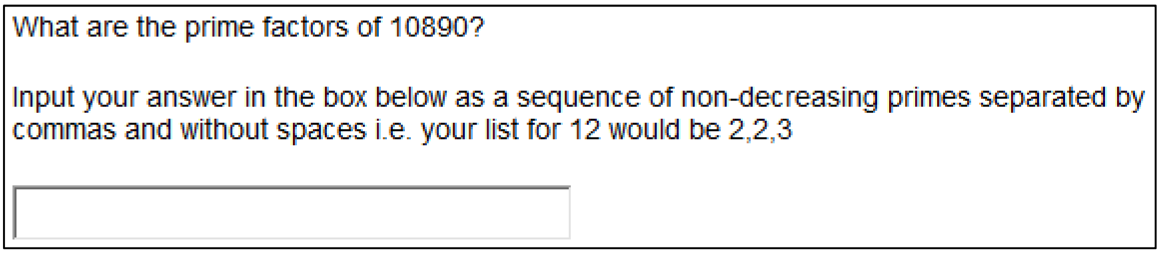


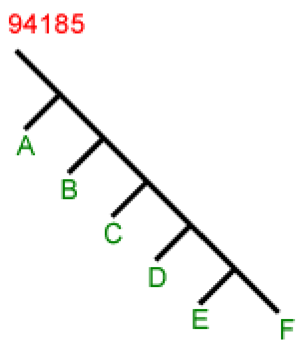
Figure 3.1: Question on listing prime factors of a number.

3.1.1. Assumed and tested skills

It is assumed that students know what the prime numbers are. A **prime number** is defined as 'an integer greater than 1 whose only positive factors are itself and 1' (Epp, 2010, p 103). The first few prime numbers are: 2, 3, 5, 7, 11, 13, 17, 19, 23, etc. On the other hand, a number d is a **factor** of a composite number $n > d$ if $d \neq 1$ and if d divides n (i.e. $d \mid n$ or n equals a multiple of d).

In three questions, students are tested on finding the **prime factors** (e.g. Figure 3.1). They are required to list all prime numbers which divide the randomly chosen number. A possible way of showing them is to use the **factor tree**. One of the questions (Figure 3.1.1) tests whether students can fill the indicated, missing values in and therefore it is assumed they know what the factor tree is and how it should be constructed.

Eileen has correctly drawn the following factorisation tree for the number 94185



Enter a value in each box for the corresponding letter in the diagram:

A = B = C =

D = E = F =

Note: These numbers should be in non-decreasing order and you can have repeats i.e. A≤B≤C≤D ...

Figure 3.1.1: An XNI question on finding prime factors of a number using factor tree.

For two other questions, on the **Highest Common Factor** (HCF), also called Greatest Common Divisor (GCD), and the **Lowest** (or Least) **Common Multiple** (LCM), it is assumed that students know how to do prime factorisation of the given number or can use Euclid's algorithm. Given two randomly-generated numbers, they are tested on finding HCF and LCM.

3.1.2. Objective questions

Word input type questions

One way of fully testing students' ability on finding prime factors of a number is to ask them for their list. To allow system correct marking, students are given short instructions on the required format of the answers (Figure 3.1).

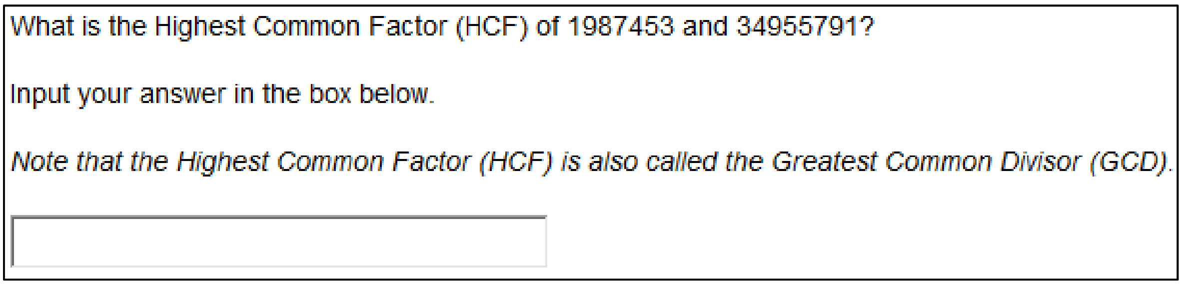
Multiple Numerical input type question

The second method to ask for the prime factors involves the factor tree. Students are required to type each of the prime numbers in the boxes corresponding to a specific numbers indicated at the factor tree presented in the question (Figure 3.1.1). The question also includes a note on ordering of the inputs.

The question presents factor trees of different sizes. This depends on the randomly chosen number and therefore on the number of prime factors of it. This then results in a varying number of input boxes with assigned alphabet letters A, B, C, etc. We call this an XNI question.

Numerical input type question

Questions on finding the HCF and the LCM are of the numerical input type. In the input box provided, students have to type values they were asked to find.

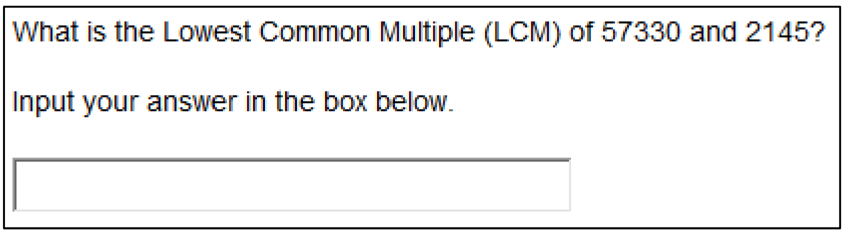


What is the Highest Common Factor (HCF) of 1987453 and 34955791?

Input your answer in the box below.

Note that the Highest Common Factor (HCF) is also called the Greatest Common Divisor (GCD).

Figure 3.1.2a: Question on finding the Highest Common Factor.



What is the Lowest Common Multiple (LCM) of 57330 and 2145?

Input your answer in the box below.

Figure 3.1.2b: Question on finding the Lowest Common Multiple.

Different styles

The database comprises 2 questions of the WI type. One of them is presented by Figure 3.1 and asks for the list of prime factors of a given number. Whereas, the other (shown by Figure 3.1.2c below), requires the full list of prime numbers being the possible divisors of any number from the range given by two random numbers.

> Gemma wants to write program to determine whether or not a randomly chosen number between 311 and 343 is prime. Gemma knows this requires checking divisibility of this randomly chosen number by a list of primes. What is that list?
>
> Input your answer in the box below as a sequence of increasing primes separated by commas and without spaces i.e. your list will start with 2,3,5, ...

Figure 3.1.2c: Question on listing prime factors of any number from the range of two given numbers.

Random parameters

All the questions have numbers involved which are chosen at random. Two questions (Figure 3.1.1 and Figure 3.1.2c) have also names selected at random but this is only a surface effect that is not changing the mathematical meaning of the question.

3.1.3. Feedback

All the feedback screens in the database start with the original questions, followed by the submitted answer. Then, depending on the correctness of the given answer, the correct answer together with the explanation follows in the cases when a wrong answer is submitted. There is a number of possible feedbacks for display, depending on the question.

One method of finding the right answer to the question presented in the Figure 3.1.2c is to firstly realise that only the highest number should be taken into account when determining the unique list of prime numbers satisfying any number from the given range. The number of possible prime numbers that should be checked is fairly large as the number is large. Therefore, listing all the prime numbers smaller than the number in the question would be a long process. However, there is a way that is presented to students on the feedback screen to limit their quantity. It involves finding the square root of the higher number and testing only the prime numbers less than this calculated value. The feedback is then completed with the explanation of the square root concept.

The feedback for the second WI type question and the XNI type question presents one method of finding the list of prime factors of the number. However, just before the solution is shown, students are given the definition and a few examples of prime numbers. The method is based on consecutive attempts of dividing the given number, or the successful division results, by different prime numbers, starting at the smallest prime and working upwards. What follows from there is a completed factor tree.

Feedback for the other two questions, on the HCF and the LCM, uses Euclid's algorithm to find the HCF and presents it in the table. Then this result is further used in the formula for the LCM as presented in Figure 3.1.3.

~~~~~~Your result~~~~~~

Your answer 23546, should have been 630630

You should have used the formula $\frac{a \times b}{\gcd(a,b)}$. Using Euclid's algorithm to find the GCD gives the table

LARGER VALUE	/	SMALLER VALUE OR REMAINDER FROM PREVIOUS LINE	=	INTEGER PART OF THE DIVISION		REMAINDER
57330	/	2145	=	26	rem	1560
2145	/	1560	=	1	rem	585
1560	/	585	=	2	rem	390
585	/	390	=	1	rem	195
390	/	195	=	2	rem	0

So GCD=195 and thus LCM=(57330 × 2145)/195=630630

Figure 3.1.3: Feedback of the question on finding Lowest Common Multiple.

3.1.4. Technical content

The coding below is shown on the example presented in Figure 3.1.2c and this is as follows:

- Selection of two numbers for the range

```
primes = new Array("",2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,
89,97,101,103,107,109,113,127,131,137,139,149,151,157,163,167,173,179,181,191,193,
197,199,211,223,227,229)

num_prime = Math.ceil(4*Math.random())+6;
range=primes[num_prime+1]*primes[num_prime+1]-primes[num_prime]*primes[num_prime];
floor = primes[num_prime]*primes[num_prime];

number1 = floor + Math.ceil(range/2*Math.random());
number2 = floor + Math.ceil(range/2*Math.random()+range/2)-1;

number1_presented = Math.min(number1,number2);
number2_presented = Math.max(number1,number2);
```

primes is an array storing prime numbers.

Math.random() returns a random number greater than or equal to 0 and less than 1, then '4*' multiplies random number by 4 and therefore number greater than or equal to 0 and less than 4 is returned. After that the ceiling function, *Math.ceil*, rounds it upwards to the nearest integer. Then the number is increased by 6 and therefore *num_prime* is a number between 7 and 10.

range is a number generated by the difference between two squares of two consecutive prime numbers with the first defined by *num_prime*.

floor is a number whose value equals the selected prime number squared.

number1, as well as *number2* are numbers generated using the values of *floor* and *range*.

Math.min() chooses the smallest value from the list of two numbers, whereas *Math.max()* the largest. Then those numbers are used to indicate the range of numbers presented to students in the question.

- Defining the correct answer

```
list_of_primes = new Array();
for(i_num = 1; i_num <= num_prime; i_num++){
list_of_primes[i_num-1] = primes[i_num];}
Correct%QUESTION.NUMBER% = list_of_primes;
```

list_of_primes is an array holding list of prime numbers being a correct answer. Its content is built using a 'for loop'. It starts with *i_num*=1, increases by 1 each time the script is run, and runs as long as *i_num* is less than *num_prime*, being a number of prime factors. At each step the prime factor of index *i_num* (*primes[i_num]*) is assigned to the *list_of_primes* array.

- Generation of the factor tree

For the description of the coding on factor tree please see the work by Park (2010).

3.2. Modular arithmetic

"The most practical use of modular arithmetic is to reduce computations involving large integers to computations involving smaller ones" (Epp, 2010, p. 483). The database comprises three questions, each with different arithmetic operations, i.e. addition, subtraction and multiplication. Therefore, questions are of different style, but all are of the numerical input type.

An example of a question on modular arithmetic is presented in Figure 3.2 below.

> Consider two natural numbers *x* and *y*.
>
> If *x* mod 96 = 77 and *y* mod 96 = 75, what is (*x*+*y*) mod 96?

Figure 3.2: Question on modular arithmetic checking knowledge of the relation $(a+b) \equiv (c+d) \bmod n$

3.2.1. Assumed and tested skills

It is assumed that students know the notation of the **modulo *n***, i.e. $a = x \bmod p$. They are expected to know that a nonnegative integer remainder *a* is obtained when *b* is divided by *p*. When *p > 0* then $x = qp + a$ where *q* and *a* are integers and $0 \le a < p$.

When a, b, x, y, p are integers with $p > 0$, and $a \equiv x \bmod p$ and $b \equiv y \bmod p$, then four rules hold:

$$(x + y) \equiv (a + b) \bmod p$$

$(x-y) \equiv (a-b) \bmod p$

$xy \equiv ab \bmod p$

$x^m \equiv a^m \bmod p$ for all positive integers m.

Students are tested on their knowledge of the first three relations that should be used to simplify calculations.

3.2.2. Objective questions

Numerical input type question

Questions are of the numerical input type. In the input box provided, students have to type values they were asked to find.

Different styles

In the questions students are given two 'modulo *n*' equalities, each having one value not known, i.e. *x* and *y* (see Figure 3.2). Then questions ask either for $(x+y) \bmod p$, $(x-y) \bmod p$, or $(xy) \bmod p$.

Random parameters

All the numbers involved in the question are chosen at random.

3.2.3. Feedback

The first information on the feedback screen is the question itself, followed by the submitted and correct answers, as well as explanation of the solution (see Figure 3.2.3). This starts with different forms to represent given equalities using the definition of modulo *n* and leads to the description of the relation being tested.

~~~~~~Your result~~~~~~
Your answer 33, should have been 56.

Observe that *x* = 96m + 77 and *y* = 96n + 75 for some natural numbers m, n.

Then *x*+*y* = 96×(m+n) + (77+75)
The first term is clearly divisible by 96 so all we need to find is (77+75) mod 96 = 152 mod 96 = 56

Figure 3.2.3: Feedback to the question presented in Figure 3.2.

3.2.4. Technical content

The coding below is shown on the example presented in Figure 3.2.3 and this is as follows:

- Selection of three random numbers to complete modulo equations

*p = Math.ceil(50*Math.random())+50;*
*a = Math.floor((p-10)*Math.random())+10;*
*b = Math.floor((p-10)*Math.random())+10;*

Values of *p*, *a*, and *b* are used to form two modulo p equalities. Before they are substituted they would be $x \bmod p = a$ and $x \bmod p = b$. Functions *Math.ceil()*, *Math.random()* (described in Section 3.1.4), and *Math.floor()* are used to randomly generate 3 values changing every time question is run. *Math.floor()* rounds the number downwards to the nearest integer and returns it.

- Defining the correct answer

Correct%QUESTION.NUMBER% = (a+b) % p;

$(x+y) \equiv (a+b) \bmod p$ rule is used to calculate the correct answer. In JavaScript *%* represents modulus operator.

3.3. Non-decimal Arithmetic

In everyday situations we use decimal numbers with 10 as the base, e.g. $379 = 3\times10^2 + 7\times10^1 + 9\times10^0$.

However, bases other than 10 are also used, e.g. in computer science, and were subject to constructing 28 questions divided into four groups: conversion of numbers between different bases, binary arithmetic, octal arithmetic, and hexadecimal arithmetic.

Since the require input is a binary string, these questions are of word input type, but are of different *styles*. The conversion of numbers to different bases involves bases 2, …, 10, and 16. The arithmetic questions include addition, subtraction, multiplication, and division of binary, octal as well as hexadecimal numbers.

An example of a question on non-decimal arithmetic is presented in Figure 3.3 below.

Claudia has to work out the following binary addition:

$11111011_2 + 10101111000_2$

What answer should Claudia give?

Please input your answer in the box below.

$\boxed{}_2$

Figure 3.3: Question on addition of two binary numbers.

3.3.1. Assumed and tested skills

It is assumed that students know that **binary notation**, also called base 2 notation, uses 0's or 1's. 8 is another base and the base 8 notation is called **octal notation**. Eight different digits from 0 to 7 are used to represent numbers. Also the base 16 expansion of an integer is used in questions involving arithmetic. It is called **hexadecimal notation** and uses sixteen different digits, i.e. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F, with A to F representing respectively decimal numbers 10 to 15.

A set of 8 questions tests students' ability on **converting numbers** between different bases. While converting decimal numbers to numbers of any other than 10 base, students should use the same method (see Figure 3.3.3a). It is based on consecutive dividing the decimal number by the base number, making a note of the remainders, until the number that should be divided next is less than the base and therefore it is the last remainder. The final step is to write the remainders starting from the last one with the first remainder being the last digit of the number to the requested base.

There is only one way of converting a non-decimal number to decimal notation. Students should see each number as a string of digits and each digit's position indicates the power of the base by which it is multiplied (Epp, 2010). Like in the example of 379 decimal number in Section 3.3 and part of the feedback in Figure 3.3.3b, the right-most digit should be multiplied by the base to the power 0, and then the powers increase by one for every digit to the left. Lastly, all the products should be added to give a decimal number.

In contrast, two questions ask for the conversion from binary notation to either octal or hexadecimal notation, and two ways of finding a solution are possible (see Figure 3.3.3b). Therefore, only the ability of conversion is tested but not the method. The harder way is to first find the decimal expansion of the binary number (described above) and then converting it to base 8 or 16 (also described above). The easier method is to group the binary number in groups of three (for octal notation) or four (for hexadecimal notation), adding zeros at the beginning if necessary. Then each block should be converted into its corresponding octal/hexadecimal representation, e.g. binary number 101 represents 5 in octal notation, whereas binary 1010 is equivalent to hexadecimal number *A*.

Questions on binary, octal, hexadecimal arithmetic are similar to each other. In each group students are tested on **addition, subtraction, multiplication and division** of two numbers of the same base. Where necessary, it is assumed that they know how to use **tables for addition and multiplication** as they are given in the question. Each group of questions includes one problem on **complements** and the ability of finding them is assessed.

3.3.2. Objective questions

Numerical input type questions

All questions are about numbers. Hence, whether they ask for the conversion of the number to different base, or some arithmetic is required, they all are of the numerical input type.

Different styles

Questions on arithmetic are clones of each other. Operations change as well as bases of the numbers. Questions on conversion differ by the bases. Separation of all these cases allows identification of the tested skills. This would not be possible if the combinations of bases or operations resulted from completely random selection.

Random parameters

Numbers in the questions are chosen at random. In most cases bases of the numbers are predefined except two questions on conversion. Their bases are between 3 and 9. At the start of the questions, where a name is used, these are also chosen at random but do not change the mathematical meaning of the problem. As mentioned previously, this is known as a ‘surface effect’.

3.3.3. Feedback

The feedback on the conversion from decimal number to a number of different base is well explained and laid out (Figure 3.3.3a). It describes the method of solving the problem (see Section 3.3.1) using numbers from the question. It presents successive division by the base number in the table of the required size.

~~~~~~Your result~~~~~~
Your answer 564314366, should have been 65131315.

To convert decimal numbers to base 7 numbers, you need to divide successively by 7; for this example this gives:

5554015/7 =	793430	remainder	5
793430/7 =	113347	remainder	1
113347/7 =	16192	remainder	3
16192/7 =	2313	remainder	1
2313/7 =	330	remainder	3
330/7 =	47	remainder	1
47/7 =	6	remainder	5
6/7 =	0	remainder	6

Now read the remainder column in red from the bottom up - hey presto, this gives the base 7 representation of the number 5554015
This is not magic - think about WHY this works and ask your teacher if necessary!

Figure 3.3.3a: Feedback of the question on conversion of decimal number to base 7 number.

Detailed feedback also accompanies questions on the conversion of the binary notation of a number to either octal or hexadecimal expansion (Figure 3.3.3b). As described in Section 3.2.3 there are two possible ways of solving these questions and they are presented to students. The first method is also a part of the feedback for the remaining questions on conversion of numbers between bases.

~~~~~~Your result~~~~~~
Your answer 56B, should have been CA.

To understand the base 2 number 11001010 , we can write it *in reverse* as:

$$0 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 + 0 \times 2^4 + 0 \times 2^5 + 1 \times 2^6 + 1 \times 2^7$$

Now we could work this out as a decimal number, and then convert it to a hexadecimal number ... but fortunately there's an easier way, as follows:

Firstly write down the binary representation of the hexadecimal numbers
0 = 0000, 1 =0001, 2 = 0010, 3 =0011, 4 = 0100, 5 = 0101, 6 = 0110, 7 = 0111, 8=1000, 9=1001, A=1010, B=1011, C=1100, D=1101, E=1110, F=1111.

Secondly, group the digits of the given number into 4's, adding zeroes at the start if needed. In this case we have 11001010 = 1100,1010,

and finally equate each group of 4 with its hexadecimal equivalent, giving the result required.

Figure 3.3.3b: Feedback on the conversion from binary number to hexadecimal number.

With one exception, feedback is not available for the questions on arithmetic. It is due to technical issues. The main difficulty is to present values that need to be carried or borrowed across columns. Well explained feedback for the question on addition of two binary numbers includes laid out arithmetic together with the description on how it should be performed (Figure 3.3.3c).

~~~~~~Your result~~~~~~
Your answer 1001110101, should have been 11001110011.

The addition of two binary numbers 11111011 and 10101111000 should be laid out as follows:

- both numbers written underneath each other (as you would align the numbers if you were adding decimal numbers)
- two lines below them, leaving some space between them to allow some writing
- the result of addition is written between these two lines
- carries are written under the bottom line (any that are 0 can be omitted)

This gives:

```
          1 1 1 1 1 0 1 1
    1 0 1 0 1 1 1 1 0 0 0
+ _ _ _ _ _ _ _ _ _ _ _ _
    1 1 0 0 1 1 1 0 0 1 1
  _ _ _ _ _ _ _ _ _ _ _ _
  0 0 1 1 1 1 1 1 0 0 0
```

with the correct answer **11001110011** (located between two horizontal dashed lines in the above solution).

Figure 3.3.3c: Feedback of the question on the sum of two binary numbers.

3.3.4. Technical content

The coding below is shown on the example presented in Figure 3.3.3a and this is as follows:

- Random selection of numbers and calculation of the correct answer

*base=Math.ceil(7*Math.random()+2);*
n=8; //denotes the max number of places to be displayed i.e. base^n
*A = Math.ceil(Math.pow(base,n)*Math.random())+20;*
B = A.toString(base);
C= parseInt(B,base);
Correct%QUESTION.NUMBER% = B;

base tells to which base the number *A* will be converted and is between 3 and 9 inclusive. It is generated using functions *Math.ceil()*, *Math.random()*, and *Math.power()*. *Math.power(base,n)* is used to calculate the value of *base* raised to the power of *n*.

To calculate the correct answer, function *toString()* on a number is used. It converts the decimal number (i.e. *A*) to the number in the randomly selected base (i.e. *base*).

In different cases, when the question asks to convert a non-decimal number to a decimal number the function *parseInt()* is used. It has two arguments; the number in the string to be converted and the base.

- Generation of the table for the feedback

```
a=A;

workings="<table border=2 cellspacing=2 cellpadding=2 >";
for (var places=0; places<n; places++)
      {if(a!=0){temp=Math.floor(a/base);
       workings=workings+"<tr bgcolor=#add8e6><td>"+a+"/"+base+" = "+"</td>
<td>"+temp+"</td><td> remainder</td> <td> <font color='RED'>
"+custRound(base*((a/base)-temp),0)+"</td><tr>";
       a=temp;}else{} }
workings=workings+"</table>"
```

The coding starts with assigning the value of a selected number for conversion to *a*. Then the table is generated at a specified border thickness, spacing between cells, as well as cells' padding size.

'for loop' (described in Section 3.1.4) executes code using 'if… else' statement. This however executes another code when the number to be divided is not 0 and does nothing if it is 0. Every time the code runs, it calculates how many times the value of the base goes into the number for division (i.e. *temp*). It also fills the table in with division of two numbers, an equals sign, the value of *temp*, the word remainder and the calculated remainder value. At last, a new value is assigned to *a*, that is the value of *temp*.

The addition of two random decimal numbers A and B are converted to any base (in this case 2) by using a *JavaScript* in-built function i.e. *C= (A+B).toString(base);* . So while the question coding is trivial, the feedback requires the layout of the addition to be placed in a borderless table to ensure alignment of the place notation and smaller numbers representing the carries below the answer line

3.4. Scientific Notation

Scientific notation is a method that is particularly useful for expressing numbers whose absolute value is very large or very small. The database comprises nine questions of either two numerical inputs type or responsive numerical input type, and of different style. Problems involve numbers in decimal notation to be expressed in scientific notation, and vice-versa. Others ask students to perform some operations on numbers.

An example of a question on scientific notation is presented in Figure 3.4 below.

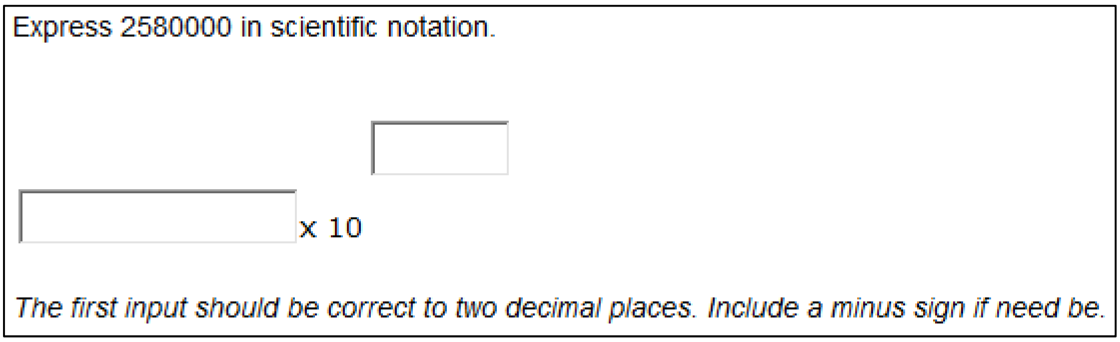


Figure 3.4: Question on expressing decimal number in scientific notation.

3.4.1. Assumed and tested skills

It is assumed that students are familiar with the format of the number in **scientific notation**, that is $a \times 10^b$, where $1 \leq |a| < 10$ and $b \in \mathbb{Z}$.

It is also assumed that participants know how to round numbers to a specific number of **decimal places** and **significant figures**. They should remember that they may need trailing zeros to indicate required accuracy, e.g. 3.50 is not the same as 3.5 when accuracy is being claimed.

Four questions test whether students can **convert numbers** between decimal notation and scientific notation. The way to do it is to move the decimal point either to the left or to the right and to calculate the power of 10. The general rule to do this is to decrease the power of 10 index by the number representing the number of places the decimal point must be moved to the right, or increase it when decimal point shifts to the left. The full explanation of the rule is presented in the Figure 3.4.3 as part of the feedback presented to students.

The remaining questions test **arithmetic** on numbers in scientific notation. Each question tests one of the following operations: multiplication or division of two scientific numbers, square root of a scientific number, as well as exponentiation of the scientific number with power being either an integer or a rational number.

3.4.2. Objective questions

Two numerical inputs type questions

Since questions ask for an alternative representation of a number or some arithmetic to be performed the best way to test the knowledge is to ask students to type the answer. Scientific numbers are written in the form of $a \times 10^b$, base *a* multiplied by 10 raised to the power of *b* being an integer. Therefore, with both *a* and *b* changing and depending on the number presented in the question, students have to find their values and type in two input boxes provided.

Responsive numerical input type questions

Questions ask for a decimal representation of a number that is given in a scientific notation form. One input box is available to students and the distractors are used to address students' answers when they used wrong power of *b.*

Different styles

Although all the questions use the same format for the answer, they provide students with different information. Some questions include one decimal/scientific number whereas others have two. They also differ by transformations and operations to be performed, as well as instruction on whether number should be rounded to certain number of decimal places or significant figures.

Random parameters

All the numbers used in the questions are randomly selected. These are decimal numbers, scientific numbers, powers of 10, integers and rational numbers to which the scientific number must be raised, the number of decimal places or significant figures required for rounding. Five questions randomly choose one of two possibilities for accuracy type.

3.4.3. Feedback

The beginning of the explanation of the question and how it should be answered is the same for every question. It includes theoretical information related to the topic. Terms of scientific notation and scientific number are introduced. Then described is the process of converting decimal number into a scientific number. Included is also note and examples on rounding.

Feedback specific to the question presented in the Figure 3.4 involves the explanation on how the decimal point should be moved to result with correct answer.

Solutions of the questions on arithmetic include the step by step calculations accompanied by the explanation of them (Figure 3.4.3). This is however missing in the questions when the number must be raised to the power being a rational number as for this question there is no standard approach of solving it.

Evaluate 4.1759 x 10^{-1} divided by 3.39701 x 10^{3}

Input your answer in scientific notation, with minus signs where necessary. The first input should be correct to 3 significant figures.

~~~~~~Your result~~~~~~

Your answers were wrong. 1.23 x 10^{-4} was required.

Scientific notation is a method of expressing very large or very small numbers.

A **scientific number** comprises two parts:

- a number which absolute value is greater than or equal to 1 and less than 10 and
- a power of 10 index.

The exponent of 10 is the number of places the *decimal point must be shifted* to give the number in scientific notation:

- If the number is 10 or greater, the decimal point has to move *to the left*, and the power of 10 will be *positive*.
- If the number is smaller than 1, the decimal point has to move *to the right*, so the power of 10 will be *negative*.

Take care when rounding and note that decimal places and significant figures may need trailing zeroes to indicate their accuracy specification e.g. 2.10 is not the same as 2.1 in terms of accuracy being claimed.
Note also that 2.10 has 3 significant figures, but only 2 decimal places.

Result	Explanation
$(4.1759 \times 10^{-1}) \div (3.39701 \times 10^{3})$	*original expression*
$= (4.1759 \div 3.39701) \times (10^{-1} \div 10^{3})$	*the terms are grouped so digit terms are next to each other followed by the power of 10 index*
$= 1.229287 \times 10^{-4}$	*the digit terms are divided in the normal way and the exponents (powers of 10) are subtracted*
$= \mathbf{1.23 \times 10^{-4}}$	*number 1.229287 is converted to scientific notation according to the accuracy specification, i.e. 3 significant figures, giving the correct scientific notation*

Figure 3.4.3: Feedback of the question on division of two scientific numbers.

3.4.4. Technical content

The coding below is shown on the example presented in Figure 3.4 and this is as follows:

- Selection of numbers: the first part of the scientific notation, exponent of the base 10, as well as decimal number equivalent to the scientific number, both being part of the question

No matter if the question asks for the conversion of decimal number to one of different base or vice versa, the numbers are coded in the same order.

```
A = (1+9*Math.random()).toFixed(2);
B = Math.ceil(2+5*Math.random());
num = Math.round(A*Math.pow(10,B));
```

A is coded first; it is the value of the first part of the scientific notation. The function used but not yet described is *toFixed().* It has two arguments (number and required accuracy) and converts a number into a string with a specified number of decimal places (i.e. 2, as it is the required accuracy). *B* is not dependent on *A* and takes values from 3 to 7 inclusive. It is the exponent (also called power or index) of the base 10 of a scientific number. *num is d*ependent on *A* and *B.* It calculates $A \times 10^{B}$ giving a decimal number, *A* and *B* being the answers to the question in Figure 3.4.

- Generation of the feedback showing how the decimal point moves to the left

```
for(iB = 1; iB <= B; iB++){
num2 = custRound(num/10,2);
if(iB == 1){Feedback%QUESTION.NUMBER% += num+" = "+num2+" &times; 10
<sup>"+iB+"</sup><br>";}else{Feedback%QUESTION.NUMBER% += num+"&times; 10
<sup>"+(iB-1)+"</sup> = "+num2+" &times; 10 <sup>"+iB+"</sup><br>";}
num = num2; }
```

The ‘for loop’ (described in Section 3.1.4) executes *num2* being a number of shifts of the decimal point by one to the left and ‘if… else’ statement. This executes the code showing how the decimal point should be moved and the power of 10 increased by one every time decimal point moves to the left. It uses HTML codes: *×* to represent multiplication operator ‘x’ and ‘[…]’ for the superscript.

4. Sets

According to Koshy (2004, p. 68) “a set is a collection of well-defined objects, called *elements* (or *members*) of the set”.

Sets are usually denoted by capital Latin letters, and membership is expressed by the symbol $\in$. Two special sets can be distinguished. One of them contains all the objects under consideration and is called the *universal set U*, whereas, the other has no elements and is called the *empty set* (denoted by the symbol $\emptyset$ or {}).

Any collection of objects can be described using the number of properties specific to the given set. Furthermore, sets are subject to many operations like A^c or $A \cup B$ etc.

A selection of properties and operations on sets form nine subtopics of this chapter. Each of them is organised in the same way, including four sections: assumed and tested skills, objective questions, feedback, and technical content.

4.1. Elements

Epp (2010) identifies two main ways of describing sets. One of them is to use the set-roster notation. It is to list all the members of a set between the curly brackets. For example, {1, 2, 3} is the set whose elements are 1, 2, and 3. With larger sets ellipsis ‘…’ can be used. It is used when a general pattern occurs, e.g. {7, 8, 9, …, 50} denotes the set of all positive integers between 7 and 50 inclusive. On the other hand, ellipsis can be used to describe countably infinite sets, e.g. {1, 2, 3, …} is the set of all positive integers.

The second method to describe a set is to use the set-builder notation. For instance, the set {1, 2, 3} in the set-builder notation can be denoted as $\{x \in \mathbb{Z}^+ \mid x<4\}$. It should be read as ‘the set of all positive integers x such that x is less than 4’, since the left part of the bracket is read as ‘the set of all’ and the vertical line as ‘such that’.

The database comprises of 18 questions on listing the elements of a set in the set-roster notation when the set-builder notation of a set is given. Questions are of word input type, but of different style. They all involve number sets, with the universal set being either a set of natural numbers, integers, or a set with a finite number of elements. The questions also differ by the number and type of inequality sign characterising the set, as well as the possible presence of the modulus symbol ‘|…|’.

An example of a question on elements is presented in Figure 4.1a below.

With the set of natural numbers is defined as

$\mathbb{N} = \{1,2,3\ldots\}$

input the following set explicitly with elements in numerical order:

$A = \{x \in \mathbb{N} \mid -3 < x \leq 2\}$

A =

Important: input your answer without spaces, separating each element with a comma and in ascending order e.g. {11,12,13}.

If you think the set has an infinite number of elements, input

- 5 consecutive elements at the start e.g. {7,8,9,10,11...},
- or 5 consecutive elements at the end e.g. {...-1,0,1,2,3},
- or 5 consecutive elements centred around 0 e.g.{...-4,-2,0,2,4...} or 6 consecutive elements centred around 0 e.g.{...-9,-3,-1,1,3,9...}.

If you think the set is empty, input **{}**

Figure 4.1a: Question on writing the set in the set-roster notation.

On the other hand, one could ask for the set-builder notation when the set-roster notation is given. However, since there is usually more than one way of doing that, it would be very difficult to correctly mark all the provided answers.

With the set of natural numbers is defined as

$\mathbb{N} = \{1,2,3\ldots\}$

input the following set in the set-builder notation:

$A = \{3,4,5,6,7,8\}$

A =

Figure 4.1b: Mock question on writing the set in the set-builder notation.

4.1.1. Assumed and tested skills

For questions on elements it is assumed that students know that elements of the set should be selected from the **universal set**. They are also expected to be familiar with the notations of $\mathbb{N}$ and $\mathbb{Z}$ that stand for the sets of natural numbers and integers respectively. On top of that, students should know the definitions of these two sets. “An integer is a whole number (not a fraction [nor an irrational number]) that can be positive, negative, or zero” (TechTerms, 2014), therefore, $\mathbb{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$. Regarding the natural numbers’ definition, there is no general agreement and different sources give different descriptions. Some authors consider non-negative integers as a natural number (e.g. Rosen, 2007; Epp, 2010), whereas others only the positive integers (e.g. Hatch and Hatch 2006). As a result of this inconsistency students are provided with the definition $\mathbb{N} = \{1, 2, 3, \ldots\}$ that should be used throughout the questions where natural numbers are the universal set.

Throughout the questions in this subtopic’s database a number of symbolic notations was used and it is assumed that students are familiar with them.

The universal set in the set-builder notation is accompanied by the **symbol of membership** $\in$. Given, for example, ‘$x \in \mathbb{Z}$’ in the left part of the bracket, it is assumed that participants know that only integers are under the consideration in the problem when finding elements of the set.

The elements in the right side of the set-builder notation are defined by the **inequality signs**. It is expected that students know these symbols and the difference between inequalities and strict inequalities.

Four questions use **absolute value** (or modulus) of x (written $|x|$).

Since students are required to list sets in the set-roster notation, the format of the answer has to be right. Expected inputs comprise three characteristics that form a set of tested skills. All the objects forming the set have something in common, i.e. they follow a rule describing all of them (Math Goodies, 2014). It is therefore important to **enumerate** the elements of the set in some order (Grassmann and Tremblay, 1996). Since we are talking here about the list of members, it is essential to separate them by **commas** (as long as the set is not empty). To finalise and clearly indicate that this enumeration forms a set it is crucial to put **curly brackets** around the list of objects. At least one student is known to have objected to being ‘marked wrong’ when inputting a sequence, e.g. 1,2,3, rather than a set {1,2,3}.

4.1.2. Objective questions

Word input type questions

The best way to fully test students' ability on listing the elements of the set explicitly, is to ask them to do it entirely themselves. Therefore, it was decided to construct word input type questions, (instead of, for example, MC questions) allowing this. Since there are many types of sets (e.g. finite, infinite, empty set) and no general formats for them to be described, a list of instructions on the required format of possible answers is given to participants (see Figure 4.1a).

Random parameters

For questions on listing the elements of a set explicitly, all the numbers are selected at random. In two questions, where the universal set is given by a finite set, its elements are set randomly between -4 and 15. Furthermore, the number of elements in the universal set is decided randomly and varies between 5 and 7. The other place in the question where random numbers are present is the set-builder notation of the set. They are used to describe the sets by setting up boundaries for the least/greatest element of the set.

Different styles

Although all the questions ask students to list the elements of the set explicitly, they differ by some parameters (as mentioned already in Section 4.1). Within questions, three types of the universal set were used. It was either a set of natural numbers, integers, or a set with a finite number of elements. The questions also differ by the number and type of inequality sign characterising the set. Out of the four possibilities for the inequality signs, one or two were assigned to each problem. The third variant in the questions is the possible presence of the modulus symbol.

4.1.3. Feedback

Similar to the other questions in the database, feedback screens comprise the question itself, followed by the submitted answer. In cases when students gave wrong answers, the correct answer is given together with the explanation on how to obtain it.

~~~~~~Your result~~~~~~

Your answer, **{-2,-1,0,1,2}**, should have been **{1,2}**.

You have to be able to read given set as 'x is a natural number greater than -3 and less than or equal to 2', where the set of natural numbers is the set of positive integers {1, 2, 3, ...}.

Integers greater than -3 and less than or equal to 2 are -2,-1,0,1,2.

However, -2,-1,0 are not natural numbers so the required set is {1,2}

Figure 4.1.3: Feedback on writing the set in the set-roster notation.

In the example above (Figure 4.1.3) the feedback is directly linked to the question and randomly selected parameters. It starts with the explanation how the set should be read using the full name (i.e. natural numbers) of the universal set instead of the symbol ℕ and the repeated definition of what the natural numbers are. Then, the numbers in the inequalities, chosen at random, are considered alongside the universal set. In the example above, the upper boundary element is part of the universal set and it is also the largest element of the set *A*. However, as stated, lower boundary element (i.e. -2) is not a natural number, i.e. we must omit -2, …, 0. Therefore, 1 (the smallest natural number) is recognised as the smallest element of the set *A*. To finalise the solution, the correct answer is given.

4.1.4. Technical content

Technical content on listing the elements of a set will be presented in this section. All the coding will be shown on the example presented in Figure 4.1a (question) and Figure 4.1.3 (feedback) and this is as follows:

- Random selection of two boundary elements of the set

myBArray = displayarray(1, -5, 9, 1);
b = myBArray[0];
b1=b+1;
myCArray = displayarray(1, b, 9, 1);
c = myCArray[0];

myBArray is an array containing one random number between -5 and 9 generated by a more general random-number function *displayarray(n, x, y, m);* that gives *n* unique numbers between *x* and *y* with flag *m* = 1 meaning that zero is allowed if in the range, else not. *b* is a number positioned at the index 0 of *myBArray* and is the lower boundary of the set (-3 in the question). Since the first inequality is a strict inequality, a number bigger than *b* by 1 is required for the feedback (i.e. *b1 = -3+1 = -2 in this case)*.

Similarly, *myCArray* is an array but containing one random number between already chosen number represented by *b* and 9. From here, one can see that $b \leq c$. Then, *c* takes a value of selected number and it is the upper boundary of the set (2 in the example above). As the second inequality is not strict, the number *c* plays an important role in the feedback.

- Possibilities of correct answers

Since the numbers are chosen at random there is more than one option for a possible correct answer. In the question, three such cases can be distinguished.

```
if(c <= 0 || b==c){Correct%QUESTION.NUMBER% = "{}"}
```

The first possibility is to obtain an empty set (*{}* to be used as an answer). One line of code has been written for two cases when no elements can be found. It has been given that the universal set is a set of natural numbers, and that $b \leq c$. Therefore, the set is empty if *c* is less than or equal to zero, or when both numbers take the same value.

```
else if(b<=0 && c>0){
Correct%QUESTION.NUMBER% = "{";
for(ib = 1; ib < c; ib++){
Correct%QUESTION.NUMBER% += ib+","}
Correct%QUESTION.NUMBER% += (c)+"}";}
```

Note: the curly bracket { } above delimits the JavaScript condition or loop whilst “{“ or “}” forms part of the answer string.

Secondly, as in the case of the presented example, when $b \leq 0$ and $c > 0$, the correct answer starts with the opening curly bracket. For the list of elements a ‘for loop’ is used. It starts with *ib=1* (since *b* is not positive and therefore the smallest element of the set is *1*), increases by 1 each time the script is run, and runs as long as *ib* is less than *c*. At each step the value of *ib* is added to the correct answer followed by a comma. After *c-1* is reached the value of *c* is added and followed by the closing curly bracket.

```
else{
Correct%QUESTION.NUMBER% = "{";
for(ib = b+1; ib < c; ib++){
Correct%QUESTION.NUMBER% += ib+","}
```

Correct%QUESTION.NUMBER% += (c)+"}";}

In all the other cases (i.e. when *b>0* and *b* is different from *c*) the correct answer comprises of curly brackets and elements assigned in a similar way to the one just described but with the difference that the ‘for loop’ starts with the number greater than *b* by 1.

- Possibilities of the feedback

Random parameters caused three possible types of correct answers (described above). However, they bring even more options for the wording of the feedback. *Feedback4%QUESTION.NUMBER%* is the coded feedback presented in the Figure 4.1.3.

Feedback4%QUESTION.NUMBER%="<p>You have to be able to read given set as 'x is a natural number greater than "+b+" and less than or equal to "+c+"', where the set of natural numbers is the set of positive integers {1, 2, 3, ...}. <p>Integers greater than "+b+" and less than or equal to "+c+" are "+Elements%QUESTION.NUMBER%+". <p>However, "+Integers%QUESTION.NUMBER%+" are not natural numbers so the required set is "+Correct%QUESTION.NUMBER%+"";

It is one of the possible displays depending on the values *b* and *c* used as set boundaries in the set-builder notation and seven such cases are coded using the ‘if…else if…else statement’. *Integers%QUESTION.NUMBER%* is an array holding integers between the lower boundary and zero (-2, -1, 0 in our example).

```
feedback = "";
if (b==c){feedback = Feedback1%QUESTION.NUMBER%;}
else if (b<0 && b+1==c){feedback = Feedback2%QUESTION.NUMBER%;}
else if (b==-1 && c>=1){feedback = Feedback3%QUESTION.NUMBER%;}
else if (b<-1 && c>=1){feedback = Feedback4%QUESTION.NUMBER%;}
else if (c<=0){feedback = Feedback5%QUESTION.NUMBER%;}
else if (b>=0 && c==b+1){feedback = Feedback6%QUESTION.NUMBER%;}
else {feedback = Feedback7%QUESTION.NUMBER%;}
```

This allows the selection of one directed feedback subject to the random parameters.

4.2. Subsets and Set Equality

The database comprises three questions on subsets and set equality. Questions within this subtopic are of True/False/Undecidable (TFU) type, but of different styles. The instructions are the same for each of the questions. However, given sets are of different relationships to each other.

An example of a question on subsets and set equality is presented in Figure 4.2 below.

Suppose **A = {-1,1,2,4,6,8}** and **B = {-1,2,4,6}**.

Consider the following table of statements.

If you think the statement is True, type T.
If you think the statement is False, type F.
If you think the statement is Undecidable, type U.

Statement	**T, F or U ?**
$A \subset B$	
$A \subseteq B$	
$A \supseteq B$	
$A \supset B$	

Remember all inputs must be either T, F or U.
Obviously with this limited number of input choices, a randomly-guessing person could do quite well on this question. To gain a mark, you therefore need to get every input correct

Figure 4.2: Question on subsets and set equality.

4.2.1. Assumed and tested skills

It is assumed that students can read a **set-roster notation** of sets given at the first line of the question. They should understand that between the brackets are listed the elements of a set. They are also expected to know an **empty set** symbol ∅ in some questions.

A table of four statements is presented, including a number of **symbols**. It is assumed that students know the set theory symbols associated with this subtopic. The symbols used within the questions, as well as some properties are described below and this knowledge is expected to be known by students.

According to Epp (2010, p. 9), "if A and B are sets, then A is called a **subset** of B, written $A \subseteq B$, if, and only if, every element of A is also an element of B". This could also be written like A is contained in B or B contains A. Also symbolic notation has another ways of representing this dependency, i.e. $B \supseteq A$.

A special case is when one of the sets is empty. For instance, if $A = \emptyset$ and B is any set, then A is a subset of any set, i.e. $\emptyset \subseteq B$.

There are also cases when two (or more) sets have the same elements. Then we are talking about sets being **equal**, what is written symbolically $A=B$. If this is the case then it is also true that $A \subseteq B$ and $B \subseteq A$ and conversely. On the other hand, if sets A and B differ by at least one member, the relationship of such two sets can be written $A \neq B$.

Using the definition of subset and equality, the term proper subset can be defined. If the two sets are given such that $A \subseteq B$, and $A \neq B$, then it is said that A is a **proper subset** of B, symbolically written $A \subset B$ (or $B \supset A$). This means that A is a subset of B, but also that two sets are different. To be more precise, all the elements of A are also elements of B, but B must have at least one element that is not in A.

Furthermore, students are expected to realise that A and B are defined for each question independently. This is so that the relationships between the two given sets are not always true and depend on the sets provided in the question.

Within the subtopic on subsets and set equality, the ability to understand the notation and apply the definitions and properties described above is tested. Participants must use the given sets and compare the elements of two sets in order to decide on the correctness of the given statements. Although the answer is never Undecidable 'U', students may not know this. For example, they may think that there is insufficient information given to decide on the truth of the statement.

4.2.2. Objective questions

True/False/Undecidable questions

Students are presented with four statements consisting of two sets connected by one of six possible symbols ($\subset$, $\subseteq$, $\supset$, $\supseteq$, $=$, $\neq$) representing the potential relation between them. Students have to decide if the given statements are true, false or there is no clear answer. In each text box, provided along the line of the statement, users have to type one of three possible letters T, F, or U standing for true, false, and undecidable respectively. With this limited number of input choices, random guessing may result in doing quite well on this question if each correct answer is rewarded. Therefore, to get a mark, students have to get all four answers right.

Random parameters

There are a number of random parameters. Starting with the beginning of the question, sets A and B change every time the question runs. Both the number of elements in each set and the elements themselves are chosen randomly. Moving to the table of statements, the positioning of A and B varies. Sometimes A is to the left of B, whereas in other cases to the right of it. The final terms that are randomly assigned are the symbols of subsets, equality, or inequality of sets.

Different styles

The first question, out of three available, was the question with the elements occurring in sets not being controlled. It was, at most times, producing sets where neither set was a subset of the other. Hence, it was decided to distinguish two cases. One of them is when A is a subset of B and the other when B is a subset of A. This assures that students are given sets where one is a subset of another.

4.2.3. Feedback

The screen shot of the feedback screen is presented in Figure 4.2.3 below. In this set of questions students are firstly provided with the question they were given, but the table is filled in with the submitted answers. The correct answers are indicated in green, whereas wrong answers are displayed in red and accompanied by the correct answer. Further down, the relations between the two sets is described in words, followed by correct symbolic statements.

Suppose **A = {-1,0}** and **B = {-2,-1,0,2}**.

Consider the following table of statements.

If you think the statement is True, type T.
If you think the statement is False, type F.
If you think the statement is Undecidable, type U.

Statement	T, F or U?
B ⊇ A	Your input T is right
B ⊃ A	Your input F should have been T
B ⊆ A	Your input F is right
B ⊂ A	Your input F is right

~~~~~~Your result~~~~~~

You got three values right. Input1 = T, input3 = F and input4 = F . However, input2 = T .

Since B contains all the elements of A , but these two sets are not equal, A is a proper subset of B .
Therefore, the only true statements are

B ⊃ A
B ⊇ A
B ≠ A

Figure 4.2.3: Feedback of the question on subsets and set equality.

According to the information in Section 4.2.1, whenever one set is a proper subset of another one (e.g. $B \supset A$) then two other relations are true, i.e. $B \supseteq A$ and $B \neq A$.

4.2.4. Technical content

Technical content on listing the elements of a set will be presented in this section. All the coding will be shown on the example presented in Figure 4.2.3 and this is as follows:

- Random selection of set cardinalities

myNArray = displayarray(1, 0, 5, 1);
n=myNArray[0];
myMArray = displayarray(1, 0, 5, 1);

m=myMArray[0];

myNArray is an array containing one random number between 0 (set *A* will be empty in this case) and 5 (set *A* will have five members). *n* is a number positioned at the index 0 of *myNArray* and it represents the size of the set *A*. At this stage, the size of set *B* (i.e. *m*) is decided in the same way and is coded by the last two lines of the code above. Therefore, *n* and *m* (sizes of sets *A* and *B*) are random integers in the interval 0 to 5 inclusive.

- Random selection of elements

```
myAArray = displayarray( n, -3, 10, 1);
myBArray = displayarray( m, -3, 10, 1);
```

myAArray and *myBArray* are arrays containing varying number of random numbers between -3 and 10. These numbers are the elements of two sets *A* and *B* and how many of them is drawn depends on the values of *n* and *m* representing the number of elements required.

- Assignment of elements to the set *B*

```
b_set = new Array();
if(m==0){b_set = ''} else{
for(mset = 0; mset <= m-2; mset++){
b_set += myBArray[mset]+",";};
b_set += myBArray[m-1];}

bset=set_tidy(b_set);
bset_string=bset.join();
if(m==0){set_b=aset}
else if(n==0){set_b=bset}
else{set_b=set_union(aset_string,bset_string);}
bset_string=set_b.join();
if(set_b==""){bset_display="&empty;"}
else{bset_display="{"+set_b+"}";}
```

This is an extended version of the coding for the set *A* since Figure 4.2.3 shows an example when set *A* is contained or equal to set *B*. Therefore, within the coding, set *B* is built up depending on set *A*.

b_set is the starting array collecting elements for the set *B*. When *m=0*, an array is empty; when *m* has value different than zero then *b_set* is formed using the ‘for loop’ (described in Section 3.1.4) with the first element being the index 0 (arrays start at index 0) value of *myBArray* and last element indexed *m-1. set_tidy* is a function in *brunel_logic* (see Appendix E) that lists up elements of the set (*b_set* in this case) into ascending order and removes repeats. *bset* array is then changed into *bset_string,* (*a* string) by the JavaScript *join()* function.

The next three lines of coding produce the final look of the set *B* (assuring that set *A* is a subset of set *B*). It is coded using the 'if…else if…else statement' and comprises of three possibilities for the set *B* to be formed, depending on already formed set *A*. If *m* (the number of elements for set *B*) was chosen to be zero, it is changed to be the same as *n(A).* If *n=0*, the existing set *B* stays the same. In all the other cases (*n* and *m* different from zero), the elements of the sets *A* and *B* are combined together to form a new set *B* (using *union()* function from *brunel_logic*) to assure that *A* is a subset of *B.*

The role of the final two lines of the code is to give the final look of the set. An empty set will be displayed as ∅ using the Unicode '∅' and elements of every other set will be placed between curly brackets.

- Matrix of all combinations for statements in the table with entries filled in with 'F's

SPmatrix = getSPmatrix(Nrow,subjects,Ncolumn,properties);

Function *getSPmatrix*, stored in *brunel_general*, returns a matrix of 'F's (standing for false). The size of the matrix is *Nrow* by *Ncolumn*, where *Nrow* is the number of statements to be defined (4 in the example) and *Ncolumn* is the number of connectives of two sets (i.e. =, ⊂, ⊆, ⊃, ⊇, ≠). Each of the possible statements, for example *B*⊃*A*, is divided into two parts. The first one is the set to the left (i.e. *B* in this example) and the second is what remains (i.e. ⊃*A*). They are stored in two arrays, *subjects* and *properties* respectively, that are two dimensions of the matrix.

- Allocation of 'T's for entries of a matrix when certain *subject* and *property* produce a true statement, accompanied with the matching feedback

```
if (set_relationship(set_1st,set_2nd,universal_string)==2)
{SPmatrix[i][2] = "T";
SPmatrix[i][3] = "T";
SPmatrix[i][6] = "T";
```

```
feed = "Since all the elements of "+sets[set0]+" belong to "+sets[set1]+", but these two sets are not equal, "+sets[set0]+" is a proper subset of "+sets[set1]+". <br>Therefore, the
```

*only true statements are <p>"+subjects[1]+""+properties[2]+"
 "+subjects[1]+""+properties[3]+"
"+subjects[1]+""+properties[6]+""; }*

set_relationship (see Appendix E) is a function stored in *brunel_logic*. When used, it returns one of three possible outputs, i.e. 1, 2, or 3. The function returns 2 when all the elements of the first set are also contained in the second set, but the two sets are different. The coding using the second case is presented as it matches the example in Figure 4.2.3. Using the ‘if statement’, the allocation of ‘T’s is executed whenever 2 is returned for a selection of sets. Since *B* is the first set of the subject in the example, it is the first index ‘*i*’ in the matrix. Its combinations with the 2nd, 3rd, and 6th entry from the array *properties* being part of the second dimension have ‘T’s allocated and are used in the feedback.

4.3. Cardinality

The database comprises seven questions on cardinality of a set. Questions within this subtopic are of numerical input type with a possibility of word input if necessary, but of different styles. They all involve non-empty number sets. The universal set is either a set of natural numbers, integers, rational number, real numbers, or a set with elements being listed (given in a set-roster notation). The questions also differ by the number and type of inequality sign describing the set, as well as the possible presence of the modulus symbol ‘|…|’.

An example of a question on elements is presented in Figure 4.3 below.

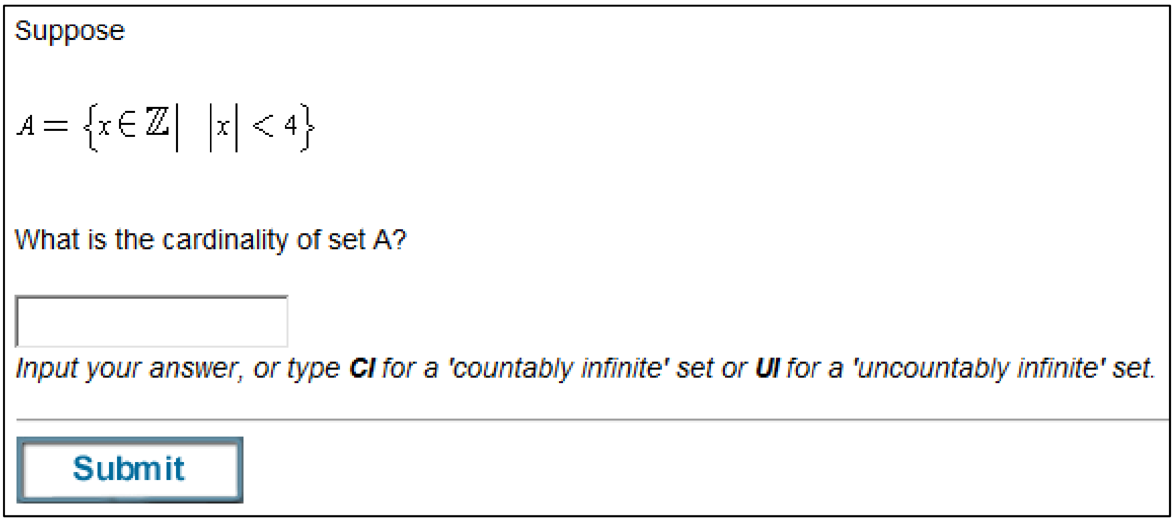


Figure 4.3: Question on the cardinality of a set.

4.3.1. Assumed and tested skills

It is assumed that students can read a **set-builder notation** of the given set. In order to do that they should be familiar with symbols including $\mathbb{N}$, $\mathbb{Z}$, $\in$, as well as inequality signs and absolute value (all introduced in Section 4.1.1). Not used in the questions on elements but present in these on cardinality are also symbolic notations of rational and real numbers being $\mathbb{Q}$ and $\mathbb{R}$ respectively. A rational number is any number that can be expressed as a fraction p/q of two integers, with the denominator q not equal to zero. Real numbers are all numbers on the number line. This includes integers and rational numbers, algebraic numbers like square roots and cube roots, and transcendental numbers like π, e etc.

Writing sets in the **set-roster notation** is a tested skill in the questions on elements. Section 4.1.1 described this process which is an assumed skill when asking students for the cardinality of the set.

The questions are testing whether students know the difference between **finite** and **infinite** sets. If the given set is finite it means that there are n distinct elements where n is a nonnegative integer and students are expected to type the number representing the number of members of the set. In all the other cases sets are infinite. Two such sets can be distinguished: **countable** and **uncountable**. Countable are not only finite sets, but any non-zero cardinality sets which elements can be placed in one-to-one correspondence with the set of natural numbers, e.g. sets of natural numbers, integers, rational numbers. On the other hand, as proven by Georg Cantor, the real numbers (or any interval of the real line of non-zero length) are uncountable (Epp, 2010).

4.3.2. Objective questions

Numerical input type questions

Questions on cardinality are asking about the number of elements in the given sets. The numerical input type questions allow direct checking of this skill in contrast to multiple choice or true/false/undecidable type questions which only allow limited possibilities for answering the question. As mentioned in the previous paragraph, some sets have infinitely many members and therefore inputs other to numbers are allowed. These are ‘CI’ for countably infinite sets and ‘UI’ for uncountably infinite sets.

Random parameters

All the numbers involved in the question are selected at random. These are numbers acting as boundaries for the least/greatest element of the set, numbers determining the size of the universal set when given explicitly, as well as numbers forming it (described in Section 4.1.2 in more detail).

Different styles

With all questions asking for the cardinality of the given set it does not differentiate the style of the questions. As mentioned in Section 4.3.1 variety of universal sets, different number and type of inequality signs, as well as a possibility of the absolute value being involved makes questions to be different from each other.

4.3.3. Feedback

All feedbacks provide the definitions of the cardinality and the number sets (except of the natural numbers as the set is defined in the question and the universal set given explicitly does not need explaining) being the universal set in the problem. Then, given is how the set should be read using necessary words for the symbols used.

If the question involves the absolute value (Figure 4.3), short information on it is provided. In the cases when the correct answer is a finite set, the set is given explicitly. Questions with real numbers as the universal set will always result 'UI' (uncountably infinite) as the answer and a note on Georg Cantor's proof is given (Section 4.3.1).

Feedback presented in Figure 4.3.3 below is used when the given set is a countably infinite subset of rational numbers. A special arrangement of rational numbers is presented allowing students to realise that every rational number will appear somewhere in the list. Furthermore, elements of any non-empty subset of rational numbers can be placed in one-to-one correspondence with natural numbers, what is the condition for the set to be countably infinite.

Suppose

$A = \{x \in \mathbb{Q} \mid \; x \leq 1\}$

What is the cardinality of set A?

*Input your answer, or type **CI** for a 'countably infinite' set or **UI** for a 'uncountably infinite' set.*

~~~~~~Your result~~~~~~
Your answer UI, should have been CI

The cardinality of a set A, denoted by |A|, n(A) or #A, is a measure of the number of elements of the set.

$\mathbb{Q}$ is a symbol denoting the set of rational numbers. A rational number is any number that can be expressed as a fraction p/q of two integers, with the denominator q not equal to zero.

In the question, you have to be able to read given set as 'x is a rational number less than or equal to 1'. Then, you have to recognise that 1 is a rational number and so is the largest element of the required set.

The smallest element is not defined.

All the rational numbers can be arranged in the following order:

$$\frac{0}{1}, \frac{1}{1}, \frac{-1}{1}, \frac{1}{2}, \frac{-1}{2}, \frac{2}{1}, \frac{-2}{1}, \frac{1}{3}, \frac{2}{3}, \frac{-1}{3}, \frac{-2}{3}, \frac{3}{1}, \frac{3}{2}, \frac{-3}{1}, \frac{-3}{2}, \frac{1}{4}, \frac{3}{4}, \frac{-1}{4}, \frac{-3}{4}, \frac{4}{1}, \frac{4}{3}, \frac{-4}{1}, \frac{-4}{3} \cdots$$

and it is clear that every rational number will appear somewhere in this list. Then, each of those numbers can be placed in one-to-one correspondence with the set of natural numbers $\mathbb{N}$ and therefore it is a countably infinite set.

Since set A is an infinite set and a subset of a set of rational numbers it must also be *countably infinite*, the correct answer should be CI.

Figure 4.3.3: Feedback of the question on cardinality.

4.3.4. Technical content

Technical content on listing the elements of a set will be presented in this section. All the coding will be shown on the example presented in Figure 4.3 and this is as follows:

- Random selection of set boundary

```
b = Math.ceil(12*Math.random())-4;          // -3 ≤ b ≤ 8
b1 = b-1;                                   // -4 ≤ b ≤ 7
```

Since strict inequality is used to define the set, *b1* is calculated to determine the highest value of the set.

- Generation of the set in the set-roster notation used in the feedback for Figure 4.3.

```
n=2*b-3;
n1=n+1;
if (b>0){
b_floor = -b+1;
Correct = "{";
for(ib = 0; ib <= n; ib++){
Correct += b_floor+ib+",";}
Correct += (b_floor+n1)+"}";}
else{Correct ="&empty;"}
```

When *b* (value of the boundary) is greater than 0, the set-roster notation of the set is build using the ‘for loop’. Preceded by the opening curly bracket, it adds one element followed by a comma to an array at each step. The first element has index ib=0 and takes value *b_floor+ib. b_floor=-b+1* since elements of the set are integers which absolute value is less than *b* and its smallest element equals to *–b+1.* This value is increased by the new index of the array while the script runs until *n=2*b-3* (index one before the last one of the array). After that the last element is added to an array followed by the closing curly bracket.

In other cases (when *b* is less or equal to 0) set is empty and Ø is displayed.

- Possibilities of correct answers

```
Correct%QUESTION.NUMBER%="";
if (b>0){Correct%QUESTION.NUMBER%=2*b-1}
else {Correct%QUESTION.NUMBER%=0}
```

In cases when *b>0* the cardinality of the set is calculated using the *2*b-1* formula and in other case it equals to 0.

4.4. Power sets

Section 4.2 is about subsets and sets equality. Examples presented there are about deciding whether one set is a subset of another one. To follow from there, all possible subsets of a particular set will be discussed here, i.e., the power set. This topic comprises six questions of different types as well as of different styles. An example of a power set question is presented in Figure 4.4.

Let A = {1,2}. Then $\mathcal{P}$(A) = {∅,{1},{2},{1,2}}.

Now let B = {1,2,9}.

What is the 6th element of $\mathcal{P}$(B) when listing elements of the power set B using the ordering shown for the set A (increasing cardinality of subsets, and elements of subsets listed in increasing numerical order)?

*Type **impossible** if you think there are not that many elements in the power set of B, or if you think it is an empty set type **{}**.*

Figure 4.4: WI type question on power set.

4.4.1. Assumed and tested skills

It is assumed that students know the set-roster notation (described in Section 4.1.1), the symbolic form of an empty set, i.e. ∅ (described in Section 4.6.1) and described already (Section 4.2), knowledge of subsets. A special fact, important to the topic of power sets, is that ∅ is a subset of every set.

Bearing in mind all these assumed skills, tested skills can now be defined. Students are tested on their knowledge of the power set defined as "given a set A, the **power set of** *A,* denoted $\wp(A)$, is the set of all subsets of *A*" (Epp, 2010, p 346). The question presented in Figure 4.4 tests students on finding the exact element of a power set, when the given set *B* has three elements. This way of asking a question eliminates a risk of making typing mistakes that would be very likely to happen if students were asked to input the full power set, but still allows testing the specific skill. The other type of question examines students' knowledge of the formula for the **number of elements of a power set** when the set is given with members listed explicitly or when only the number of elements of the set is given. The power set has 2^n elements when n is the number of elements of a given set.

4.4.2. Objective questions

Word input type questions

Similar to the questions where students had to input the elements of a set explicitly, the question asking them to input the exact element of a power set requires students to do the questions themselves without any suggestions from the questions (as in the case of MC questions). Even though participants are asked to find just one element, finding most or even the whole set in order to count the n^{th} element is necessary. Also the option of typing 'impossible' when there are not that many elements of a power set or '{}' if the answer is an empty set makes the question more demanding.

Numerical input type question

The question with a set given in a set-roster notation and asking to find the number of elements of its power set is of a NI type (see Figure 4.4.2a). In order to get it right it is enough to know the general formula above without the need to write the power set explicitly.

Suppose Q = {-9,-8,-7,-6,-5,-4,-3,-2}. How many elements does $\mathcal{P}$(Q) have?

Figure 4.4.2a: NI type question on power set.

Responsive numerical input type question

This question provides students with the information on the number of elements of a set and asks to find the number of elements of its power set. In contrast to the NI question it is impossible to list all the elements of the power set (unless doing it for a chosen set with *n* elements) and so the formula 2^n must be used. On the top of this, students get directed feedback if they inputted one of the possibilities for a wrong answer. The mal-rules for this question are: *n* i.e. the number representing the number of elements of a set itself, *2n* i.e. wrong formula, and 2^n-1 i.e. not including the empty set or the set with all the elements.

Multiple choice type question

In contrast to WI question, the MC question (see Figure 4.4.2b) doesn't fully check students' understanding of the concept. A chance of choosing an answer from the list of possible answers makes the question much easier and a lucky guess or some other thinking process may result in getting the question right. However, it builds students confidence and allows them to try easy question before moving to the harder one where they have to list all the elements themselves.

Find the power set of the set W = {7,8,4}

○ $\mathcal{P}$(W) = {{4},{7},{8}}
○ $\mathcal{P}$(W) = {{4},{7},{8},{4,7},{4,8},{7,8}}
○ $\mathcal{P}$(W) = {{4},{7},{8},{4,7},{4,8},{7,8},{4,7,8}}
○ $\mathcal{P}$(W) = {∅,{4},{7},{8},{4,7},{4,8},{7,8}}
○ None of these!
○ I don't know!

Figure 4.4.2b: MC type question on power set.

Distractors depend on an empty set and the set containing all the elements. They are either present in the power set or the set lacks one or both of them.

Different styles

The database includes 3 questions of the WI type. One question involves the set with three elements and asks for an element of the power set. Similarly for the other question, when the set has four elements. On the other hand, the third question always asks about the element that cannot be found as the power set does not contain that many elements. Thanks to the option of inputting ‘impossible’ in all of these questions prevents students assuming that since there is an option for it, this must be a correct answer.

Random parameters

In the questions when sets are given in the set-roster notation, all the elements are chosen at random. Also the sizes of them vary (except those of WI type when they are predefined to have three or four elements). The questions of NI, RNI, MC type, described above, have the letters representing sets randomly selected.

4.4.3. Feedback

Since the topic on power sets consists of questions of four different types, a selection was made and the feedback of the question in Figure 4.4.3 of the RNI type will be explained.

The specific feedback follows starts with the explanation of the symbolic notation of the power set and the definition of the term. Later, the formula to find the number of elements is given, followed by the reminder of the number of elements in the set. This is substituted into the formula and the correct answer is calculated. The whole feedback finishes with a statement of the possible reason for the given wrong answer. However, this is only possible if student used one of the three coded mal-rules described in Section 4.4.2. Figure 4.4.3 shows feedback when the mal-rule 2*n* is used: this triggers further advice not shown here.

Suppose R is a set with exactly 5 distinct elements. How many elements does $\mathcal{P}$(R) have?

~~~~~~Your result~~~~~~

Your answer, 10, is incorrect. It should have been 32

$\mathcal{P}$(R) denotes the power set of R. It is the collection of all subsets of a given set R (including the empty set and the full set R itself).

If R is a finite set with n elements, then the power set of R contains 2^n elements.

Since R is a set with exactly 5 distinct elements, the power set of R has 2^5 = 32 element(s).

Figure 4.4.3: Feedback of the RNI question on power set.

4.4.4. Technical content

A selection of the technical content of the power set will be presented in this section. Most of it describes the way of building the question in Figure 4.4, but includes how to find the number of elements in the power set.

- Random selection of elements

myAArray = displayarray(3, 0, 9, 1);
a1=myAArray[0];
a2=myAArray[1];
a3=myAArray[2];

myAArray is an array containing three random numbers between 0 and 9. *a1, a2, a3* are the numbers positioned at indexes 0, 1, and 2 of *myAArray* and the elements of a set given in the question.

- Finding the power set

```
var a_set = new Array(""+myAArray[0]+"", ""+myAArray[1]+"", ""+myAArray[2]+"");
aset = a_set.sort(function(a,b){return a - b});
aset_string=aset.join();
```

```
var power_set = new Array("\&\phi;","{"+aset_string.charAt(0)+"}", "{"+aset_string.charAt(2)+"}",”{"+aset_string.charAt(4)+"}","{"+aset_string.charAt(0)+","+aset_string.charAt(2)+"}","{"+aset_string.charAt(0)+","+aset_string.charAt(4)+"}","{"+aset_string.charAt(2)+","+aset_string.charAt(4)+"}","{"+aset_string.charAt(0)+","+aset_string.charAt(2)+","+aset_string.charAt(4)+"}");
```

a_set is an array containing the elements of the randomised set *B* in the example. Then it is rearranged ascending order using the *sort()* function (see W3Schools, 2015) and changed into a string by the *join()* function. Lastly, an array of the power set is formulated by explicit listing all its elements. This is obtained by calling consecutive characters of the *aset_string containing the elements of a set* B.

- Number of elements of a power set

```
p=Math.pow(2, b);
```

Math.pow() is a method that returns the value equal to 2^b that is the number of elements of a power set of pre-determined cardinality *b*.

- Choosing the element to be asked

```
myCArray = displayarray( 1, b+2, p, 1);
c=myCArray[0];
```

myCArray is an array containing one random number between *b*+2, that is 3+2=5 and *p* (the number of elements of a power set). *c* is the number positioned at indexes 0 of *myCArray* and is the n^{th} element of a power set that students are asked to find.

- Correct answer

```
Correct%QUESTION.NUMBER% = power_set[c-1];
```

Since arrays start at index 0, therefore, 1 is subtracted from *c*, and therefore the element positioned at index *c-1* of a *power_set* array is a correct answer.

4.5. Cartesian Product

The database comprises of three questions. Two are of numerical input (NI) type and one is of word input (WI) type. Questions also differ in a style. An example of a question on Cartesian products is presented in Figure 4.5.

Suppose A is a set with 3 distinct elements and B is a set with 4 distinct elements. How many elements does $A \times B$ have?

$n(A \times B) =$

Figure 4.5: NI question on Cartesian product.

4.5.1. Assumed and tested skills

It is assumed that students know that a **set** is a collection of objects. Later, it is assumed that students know what the symbolic notation $A{\times}B$ represents. Given two sets A and B, "the **Cartesian product** of A and B, denoted by $A{\times}B$, is the set of all ordered pairs (a,b), where $a \in A$ and $b \in B$" (Rosen, 2007).

Two questions ask to find $n(A{\times}B)$. Therefore, students are tested on finding the number of elements in the set $A{\times}B$, that can be found by multiplying the number of elements in set A by the number of elements in set B (i.e. symbolically $n(A{\times}B)= n(A){\times}\ n(B)$). This is, however, an indirect way of testing them on listing the elements of a Cartesian product of two sets. A step further is to ask them for a specific element of it (Figure 4.5.3). This allows testing students on a Cartesian product, while limiting a risk of making typing mistakes that would be likely to happen if asked for the full set.

4.5.2. Objective questions

Numerical input type questions

NI questions were designed to test indirectly the understanding of the concept of Cartesian product. Students have to input the number representing the number of elements in the Cartesian product of two sets.

Word input type questions

Asking students to give the Cartesian product would be subject to input errors, as listing a long string of characters brings a risk of making non-conceptual mistakes, i.e. missing or double typing a bracket or a comma. Multiple choice question would not fully check the knowledge as lucky guessing easily can result in the mark. On the other hand, asking for one element requires finding the whole or most of it and then counting its n^{th} element. An option for input is word 'impossible' if students think there is not that many elements in the Cartesian product, however this is never a correct answer but makes the question more challenging.

Different styles

In two questions students are given the information on how many elements there are in each set. Then, they are asked to find either $n(A \times B)$ or $n(B \times A)$. In the third question participants are presented with two sets. This is preceded by an example of a way in which the elements of a Cartesian product should be ordered and followed by the question asking for a specific element that needs to be found.

Random parameters

The parameters that are chosen at random are the numbers representing the number of elements in sets *A* and *B.* Randomised is also the size of the sets if they are part of the question and their elements. Chosen at random is the number representing the n^{th} element of a set that students are asked to find.

4.5.3. Feedback

The feedback (see Figure 4.5.3) starts with the repeated question, followed by the submitted and correct answers. After that, the explanation starts with the definition of the Cartesian product. From there follows the formula on how the number of elements of a Cartesian product should be found and the arithmetic using the size of sets *C* and *D* to show that it is possible to find the n^{th} element. At the end, a Cartesian product of given sets *C* and *D* is given from which the correct answer can be found.

Let A = {1,2} and B = {3,4}. Then A × B = {(1,3),(1,4),(2,3),(2,4)} where we have used lexographical ordering.

Now let

$C = \{0,2,4\}$

and

$D = \{4,6\}$

What is the 5th element of C × D when listing elements using the lexographical ordering shown for the A × B?

*Type **impossible** if you think there are not that many elements in a set.*

~~~~~~Your result~~~~~~
Your answer, **(2,3)**, should have been **(4,4)**.

A × B denotes the Cartesian product of two sets A and B. It is the set of all ordered pairs *(a, b)*, where a ∈ A and b ∈ B.

The number of elements in the Cartesian product of A and B is given by *n(A × B) = n(A) · n(B)* where the · means multiply.

Since C contains 3 elements and B contains 2 elements, the Cartesian product has 3 · 2 = 6 elements.

C × D = {(0,4),(0,6),(2,4),(2,6),(4,4),(4,6)}

Then counting gives the 5th element of this set as **(4,4)**

Figure 4.5.3: Feedback of the WI question on Cartesian product.

4.5.4. Technical content

Technical content on Cartesian product will be presented in this section. All the coding will be shown on the example presented in Figure 4.5.3 and this is as follows:

- Cartesian product used in the feedback

```
cartesian_set = "";
for(ip=0; ip<=n-1; ip++){
for(im=0; im<=m-1; im++){
cartesian_set+="("+cset[ip]+","+dset[im]+"),"}}
long=cartesian_set.length;
cartesianproduct="{"+cartesian_set.slice(0,long-1)+"}";
```

cartesian_set is an array containing elements of a Cartesian product of set *C* and set *D.* It is generated using a double ‘for loop’. For each value of *ip* from 0 to *n-1* (n is the number of elements in set *C*) *im* takes values from 0 to *m-1* (n is the number of elements in set *D*) and assigns elements to a Cartesian product. They are formed by opening round bracket, element from set *C*, comma, element from set *D*, the closing round bracket, and comma to separate elements of set. After all the elements of a Cartesian product are listed the final look is obtained by including the curly brackets at the beginning and end of the list of elements, as well as returning all but the last element of an array holding the elements of a Cartesian product. It is because the last element is a comma generated from a ‘for loop’.

- Correct answer

```
p=n*m;
myHArray = displayarray( 1, m-0+1, p, 1);
h=myHArray[0];
g=Math.ceil(h/m);
f=Math.floor(h/m);
k = custRound(m*((h/m)-f),0);
if(k=="0"){delement=k+m}
else{delement=k}
Correct%QUESTION.NUMBER% = "("+cset[g-1]+","+dset[delement-1]+")";
```

p is the number of elements of the Cartesian product. It is equal to the product of the number of elements in set *A* (*n*) and the number of elements in set *B* (*m*).

h is the n^{th} element from the question and takes the value between *m+1* and *p.* Therefore, students are asked to find the element that can be defined but is not one of the first few elements. This requires writing part or whole Cartesian product, unless they can work out which element from set *C* and which from set *D* form the correct answer as used in the coding.

g is the number defining which element from set *C* forms the element of a Cartesian product that students are asked to find. It uses the *Math.ceil()* function which rounds up the result of the division of *h* by *m* to the nearest integer. On the other hand, *k* defines which number from set *D* should be taken to complete the answer. It uses *custRound()* which rounds the number to the nearest integer because the second input is zero and *Math.floor()* which rounds the number downwards to the nearest integer.

Since arrays holding elements of sets *C* and *D* start at index 0 the value of *g* and *k* have to be decreased by 1. Therefore, due to the possibility of *k* taking the value of 0, ‘if statement' is used to increase *k* by the value of *m* when it is the case and leaves *k* unchanged otherwise. When executing this code *delement* is executed taking the value of *k* with or without alteration.

4.6. Union and Intersection

Union and intersection are two operations on sets. The symbols representing these operations are '∪' and '∩' respectively. The analyses of the topics as well as examples of the questions have a lot in common. Therefore, they are not separated and both are described in this section.

The database comprises of 27 questions. There are 13 questions on the union of sets and 14 questions on the intersection. Corresponding questions from these two topics are clones of each other.

An example of a question on union using the symbol to represent operation is presented in Figure 4.6a followed by a corresponding example on intersection using word 'and' in Figure 4.6b.

Let

$$A = \left\{x \in \mathbb{Z} \middle|\ x \geq 6\right\}$$

and

$$B = \left\{x \in \mathbb{Z} \middle|\ x \geq 13\right\}$$

Input $A \cup B$ explicitly with elements in numerical order:

$A \cup B$ =

Important: input your answer without spaces, separating each element with a comma and in ascending order e.g. {11,12,13}.

If you think the set has an infinite number of elements, input

- 5 consecutive elements at the start e.g. {7,8,9,10,11...},
- or 5 consecutive elements at the end e.g. {...-1,0,1,2,3},
- or 5 consecutive elements centred around 0 e.g.{...-4,-2,0,2,4...} or 6 consecutive elements centred around 0 e.g.{...-9,-3,-1,1,3,9...}.

If you think the set is empty, input **{}**

Figure 4.6a: Question on the union of two sets using union symbol to represent operation.

Two sets A and B are given by:

$A = \{x \in \mathbb{Z} | \ x \geq 13\}$

$B = \{x \in \mathbb{Z} | \ x \geq -3\}$

Set C comprises elements in A or B. Input C explicitly with elements in numerical order:

C =

Important: input your answer without spaces, separating each element with a comma and in ascending order e.g. {11,12,13}.

If you think the set has an infinite number of elements, input

- 5 consecutive elements at the start e.g. {7,8,9,10,11...},
- or 5 consecutive elements at the end e.g. {...-1,0,1,2,3},
- or 5 consecutive elements centred around 0 e.g.{...-4,-2,0,2,4...} or 6 consecutive elements centred around 0 e.g.{...-9,-3,-1,1,3,9...}.

If you think the set is empty, input **{}**

Figure 4.6b: Question on the intersection of two sets using word 'and' to represent intersection.

These two examples are clones. Both have integers as universal sets and set-builder notations include the same (i.e. greater than or equal to) inequality signs for sets *A* and *B*. However, since one uses symbol and the other word to describe operation on sets the text is reworded slightly. Furthermore, they include some aspects present in questions on elements, described in Section 4.1. These are as follows: sets are given in the set-builder notation, expected answer must be in the set-roster notation, and the questions finish with instructions on the required format of possible answers.

Within each subtopic, on union and intersection, all questions ask to find $A \cup B$ and $A \cap B$ respectively. All questions are of the WI type, however, they differ in styles and parameters (described in Section 4.6.2).

4.6.1. Assumed and tested skills

Questions on the union and intersection are similar to the questions on elements in their design, but more complex as all the assumed and tested skills described in Section 4.1.1 are now the assumed skills for the questions on union and intersection. Therefore, it is assumed that students know the meaning of the **universal set**; they know **symbolic**

notations of elements' membership (i.e. ∈), of inequalities (i.e. $<$, $>$, $\leq$, $\geq$), as well as that they are able to write sets explicitly using **enumeration**, **commas** between the elements, and putting **curly brackets** around the elements.

In the questions where sets are given in the set-roster notation it is assumed that students know what it means when, for example, $A=\emptyset$. They are expected to know that symbol $\emptyset$ represents a set with no elements, or in other words, an **empty set**. An example including this notation is presented in Figure 4.6.2.

The questions are designed to test the following skills. In general, if *A* and *B* are sets, "the ***union*** of the sets *A* and *B*, denoted by $A \cup B$, is the set that contains those elements that are either in *A* or in *B*, or in both" (Rosen, 2007). Whereas, "the ***intersection*** of the sets *A* and *B*, denoted by $A \cap B$, is the set containing those elements in both *A* and *B*" (Rosen, 2007). The way to find elements that are the union/intersection of two sets is to look at the elements of each set and carry out the **pairwise identification**. These elements that are identical for both sets are also the elements of the intersection of two sets. The union of two sets is the set including the elements of the first set and the elements from the second set that were not identical with the elements of the first one. Since some questions use **symbol** and some **words** to represent union and intersection students are tested whether they know the optional representations of two operations.

4.6.2. Objective questions

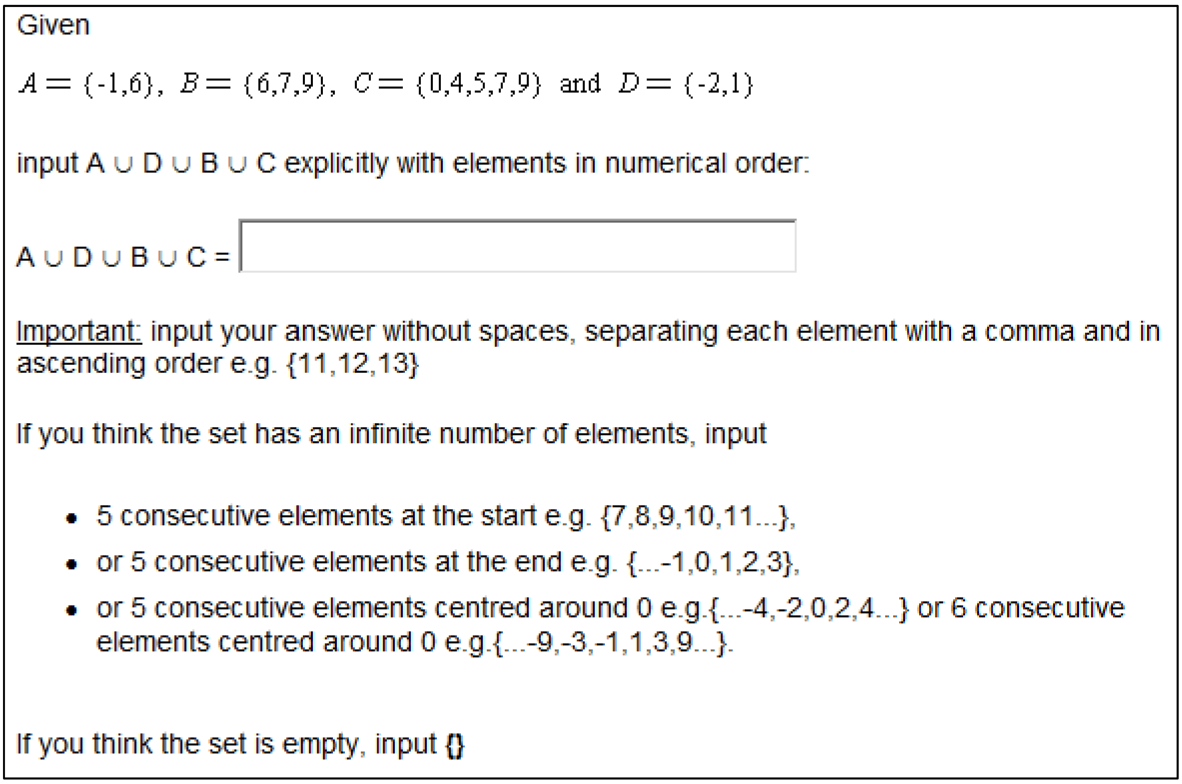
Word input type questions

Similarly to the questions on listing the elements explicitly, the questions on union and intersection of sets are of the word input type. Questions on operations are the extended version of those on elements and are of similar format. All ask for sets in set-roster notation providing a number of instructions as presented in Figure 4.6a, Figure 4.6b and described in Section 4.1.2.

Different styles

Eleven questions from the group of thirteen on the set union, as well as twelve out of fourteen on the set intersection are clones of each other. They differ either by the universal set (natural numbers in Figure 4.6.3 or integers in Figures 4.6a and 4.6b) or by the inequality signs used to describe the sets *A* and *B.* In some questions operations are described by the symbols, whereas in others they are given using words 'and' and 'or' for intersection and union respectively.

The remaining four questions (two on union and two on intersection) are of slightly different style to the other questions. Either three or four sets are given in the set-roster notation. Students are asked to find union/intersection of those sets, as presented in Figure 4.6.2 below.



Given

$A = \{-1,6\}$, $B = \{6,7,9\}$, $C = \{0,4,5,7,9\}$ and $D = \{-2,1\}$

input A ∪ D ∪ B ∪ C explicitly with elements in numerical order:

A ∪ D ∪ B ∪ C =

Important: input your answer without spaces, separating each element with a comma and in ascending order e.g. {11,12,13}

If you think the set has an infinite number of elements, input

- 5 consecutive elements at the start e.g. {7,8,9,10,11...},
- or 5 consecutive elements at the end e.g. {...-1,0,1,2,3},
- or 5 consecutive elements centred around 0 e.g.{...-4,-2,0,2,4...} or 6 consecutive elements centred around 0 e.g.{...-9,-3,-1,1,3,9...}.

If you think the set is empty, input **{}**

Figure 4.6.2: Question on the union of four sets with sets given in the set-roster notation.

Question combining union and intersection of three or four sets would not be a problem to code, however it would not be straight forward where to categorise such a question as it would be testing both operations.

Random parameters

Looking at Figures 4.6a and 4.6b, only the numbers present in sets *A* and *B* are randomly selected.

However, in the question presented in Figure 4.6.2 the sizes of the sets are randomised and may include no elements or up to 7 elements. Sets in this example have 2, 3, 5 and 2 elements. Also elements contained in them are chosen randomly and take values between -1 and 9.

4.6.3. Feedback

Tests that students are provided with for an assessment are generally ordered in the way that questions on elements appear before those on the union or intersection of two sets. Therefore, the feedback screens do not provide the information on how the set-builder notation of a set should be read and what the symbolic notations of natural numbers/integers mean. If students got the question on elements wrong, they were provided with this information already and should know that it applies here as well. On the other hand, if they got these earlier questions right but the questions on union or intersection wrong, then these assumed skills are not the problem. They are probably struggling with the concept of union/intersection. Hence, they are firstly provided with the definition of the union/intersection. After that information, the set-roster notation of the two sets is given and the correct answer follows (see Figure 4.6.3).

The set of natural numbers is defined as

$$\mathbb{N} = \{1,2,3...\}$$

Two sets A and B are given by:

$$A = \{x \in \mathbb{N} \mid x < -4\}$$

$$B = \{x \in \mathbb{N} \mid x \leq 5\}$$

Set C comprises elements in A or B. Input C explicitly with elements in numerical order:

~~~~~~Your result~~~~~~

Your answer, **{-3,-2,-1,-0,1,2,3,4,5}**, should have been **{1,2,3,4,5}**.

Set C comprises elements in A or B i.e. its elements are in the union of two sets A and B. This is denoted by $A \cup B$ and is the set of elements which belong to A or to B.

Since A = ∅

and B = {1,2,3,4,5},

$A \cup B$ = {1,2,3,4,5}.

Figure 4.6.3: Feedback on the union of two sets using word 'or' to represent union.

4.6.4. Technical content

Technical content on the union of two sets will be presented in this section. All the coding will be shown on the example presented in Figure 4.6.2. The question gives four sets in the set-roster notation. Therefore, the coding describing the process of creating sets is similar to the one described in Section 4.2.4, as the question in Figure 4.2.3 also involves sets with elements listed explicitly, and will not be repeated here. The coding for the question on the union of two chosen sets presented in Figure 4.6.2 is as follows:

- Random selection of elements

```
if(set0==1){firstset=aset_string;  firstdisplay=aset_display}
else if(set0==2){firstset=bset_string;  firstdisplay=bset_display}
else if(set0==3){firstset=cset_string;  firstdisplay=cset_display}
else{firstset=dset_string;  firstdisplay=dset_display}
```

'if…else if…else statement' is used to assign the right sets of elements, subject to the order they were chosen, e.g. A∪D∪B∪C, for the question. These particular lines of code are for the first set and the other are coded in a similar way. Depending on which condition is true (i.e. which set is chosen to be the first one) the relevant allocation takes place. Thus, for instance, if set *A* is to be selected, *firstset* is a string containing elements of the set *A* only, and *firstdisplay* is a set of elements as displayed to students.

- Finding union of two sets

```
if (a_set=="" && b_set==""){unionab = ""; unionab_string="";}
else if(a_set==""){unionab = bset; unionab_string=unionab.join();}
else if(b_set==""){unionab = aset; unionab_string=unionab.join();}
else{unionab=set_union(aset_string,bset_string); unionab_string=unionab.join();}
```

'if…else if…else statement' is also used to find the union of two sets (sets *A* and *B* in this example) depending on the true for the sets condition. *set_union* (see Appendix E) used in the last line of the code is the function in *brunel_logic.* It finds the union of two sets and returns it with elements in an alphabetical order.

- Correct answer

```
if (unionab == "" && unioncd == ""){unionabcd = ""; union_string =""}
else if (unionab == ""){unionabcd = unioncd; union_string=unionabcd.join();}
else if (unioncd == ""){unionabcd = unionab; union_string=unionabcd.join();}
```

```
else {unionabcd = set_union(unionab_string,unioncd_string);
union_string=unionabcd.join();}
if(union_string == ""){Correct%QUESTION.NUMBER% = "{}"}
else (Correct%QUESTION.NUMBER% = "{"+union_string+"}")
```

Similarly to finding the union of two sets described above, correct answer (*union_string*) is found if the set is not empty. The two sets used to find the union are union of sets *A* and *B* and union of sets *C* and *D.*

Few blocks of code are used since *set_union* function allows the performance of the union operation on two sets only.

4.7. Difference

The difference of two sets is another operation on sets next to the union and intersection of sets. It is represented by either the symbol of backslash '\' or a minus sign '-' with the first one being used throughout the questions on sets.

The database comprises 12 questions on the difference of two sets. All questions are of the word input type, but of different style. They differ from each other by the relationship of two sets, as well as the order of sets (i.e. *A\B* or *B\A*). Furthermore, some questions use symbol to express the operation on sets, whereas others the words 'but not in'.

An example of a question on the difference of two sets using symbol of backslash is presented in Figure 4.7.

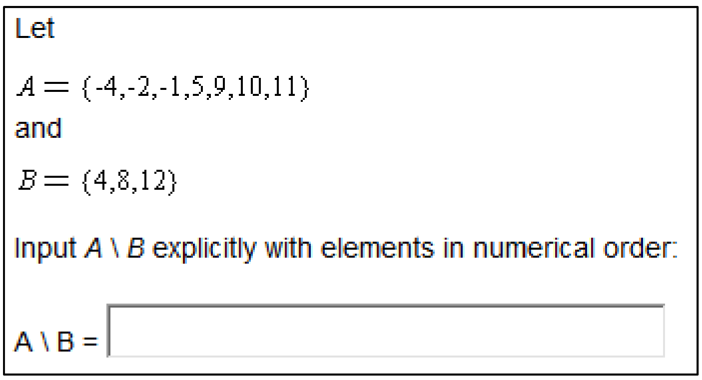


Figure 4.7: Question on difference of two sets using backslash symbol to represent difference.

4.7.1. Assumed and tested skills

It is assumed that students attempting the questions on difference possess some knowledge prior to the assessment. They are expected to know the **set-roster notation** (described in Section 4.1.1) of sets, i.e. enumerating the elements, separating them by commas, and placing everything between the curly brackets if the set is not empty. Students are also expected to know that symbol ∅ represents an empty set.

Students are tested on finding the **difference** of two sets when they are given the sets with elements listed explicitly. The difference of two sets *A* and *B* can be defined as "the set containing those elements that are in *A* but not in *B"* (Rosen, 2007). At first, it involves the **pairwise identification** of elements. Then, the answer can be stated as the difference of two sets, that is a set containing elements from the first set that were not matched with members in the second set. Word 'difference' is not used in the question therefore students are tested whether they know the optional representations (symbolic and words) of the operation.

4.7.2. Objective questions

Word input type questions

Questions on the difference of two sets are, as well as those on elements, union, and intersection, of the word input type. For the description of required formats of answers see Section 4.1.2 and Section 4.6.2.

Different styles

All questions provide students with the sets given explicitly. Some questions ask for the difference between sets *A* and *B* (see Figure 4.7), whereas others between sets *B* and *A* (see Figure 4.7.3). Each group then consists of questions with the same restrictions on sets: sets are equal to each other; the first set has more, less, or the same number of elements compared to the second set. In some questions difference of two sets is described by the symbol, whereas in others it is given using words 'but not in'.

Random parameters

Similar to questions on the union and intersection of two sets, random parameters are involved in selecting the size of the sets as well as their elements.

4.7.3. Feedback

The feedback (see Figure 4.7.3) contains the definition of the difference of two sets with the possible two symbolic notations. After that information, two sets are presented and accompanied by the method of obtaining the correct answer.

Two sets A and B are given by:

$A = \{-3,-1,1,3,4,6,12\}$

$B = \{-3,0,1,5,8,11,12\}$

Set C comprises elements in B but not in A. Input C explicitly with elements in numerical order:

~~~~~~Your result~~~~~~
Your answer, **{2,5,6,11,12}**, should have been **{0,5,8,11}**.

Set C comprises elements in B but not in A i.e. its elements are in the difference of two sets B and A. This is denoted by B \ A or sometimes by B - A and is the set of elements which belong to B but not to A.

Since A = {-3,-1,1,3,4,6,12} and B = {-3,0,1,5,8,11,12}, the elements -3,1,12 of the set B are also present in A.

Therefore, the correct answer for B \ A should be {0,5,8,11}

Figure 4.7.3 Feedback on difference of two sets using words 'but not in' to represent difference.

4.7.4. Technical content

Technical content on the difference of two sets will be presented in this section. Since sets are given in the set-roster notation and the process of creating sets is described in Section 4.2.4, it will not be presented here. Below presented will be the coding unique to the question on the difference (see Figure 4.7.3) and this is as follows:

- Defining the universal set and the intersection of two sets required for further coding

```
abunion = set_union(aset_string,bset_string);
abunion_string=abunion.join();
universal =abunion;
universal_string=universal.join();
intersection = set_intersection(aset_string,bset_string);
```

intersection_string = intersection.join();

Universal set is required for the function on finding the difference of two set (described below). Therefore, it was decided to use the *set_union* function (see Appendix E) to find the union of sets *A* and *B*, to be the universal set at the same time.

On the other hand, the *set_intersection* function (see Appendix E) is used to explain the solution in the feedback screen, as intersecting elements are those that are pairwise identical.

- Finding the difference of two sets

difference = set_difference(bset_string,aset_string,universal_string);

set_difference (see Appendix E) is the function in *brunel_logic.* It finds the difference of two sets and returns it with elements in an alphabetical order. It requires three parameters: strings of elements of the set *B* (first since the question was to find *B\A*), set *A*, and the universal set. The *set_difference* function works in a way that it finds $B \cap A^c$ (equivalent to *B\A)* and the universal set is required to find A^c (complement of *A*, described in Section 4.8).

- Correct answers for display

if(m==0){Correct%QUESTION.NUMBER% = "{}"}
else if(n == 0){Correct%QUESTION.NUMBER% = ""+bset_display+""}
else if(aset_string==bset_string){Correct%QUESTION.NUMBER% = "{}"}
else {Correct%QUESTION.NUMBER% = "{"+difference+"}"}

'if…else if…else statement' is used to execute one of four possible answers, depending on the size of set *A* and *B* (values of *n* and *m* respectively) or on the relationship of two sets (whether they are the same or different) . Information on set identities described later in Section 4.9.1 was used to code the correct answers.

- Part of the feedback

if(intersection_length == "1"){intersecting_elements = "there is one element, " +intersection+ ", of the set B that is also present in A"}

else if(intersection_length == "0"){intersecting_elements = "there are no elements of the set B that are in A"} *else {intersecting_elements = "the elements " +intersection+ " of the set B are also present in A"}*

The second sentence of the explanation to students on the correct answer includes the information on what elements are present in both sets (pairwise identical elements). This is obtained by the third condition in the code above. ‘if…else if…else statement’ executes the part of the feedback depending on the number of elements identical in both sets.

4.8. Complement

The complement of a set is the fourth operation on sets that will be described. The database comprises of 13 questions of WI type. The questions differ in style either by the questions or restrictions on the sets for which complements have to be found. Most sets involve symbolic representation of the complement with two words describing it, and not using the word ‘complement’ in either case.

An example of a question on the complement of a set using symbolic representation is presented in Figure 4.8.

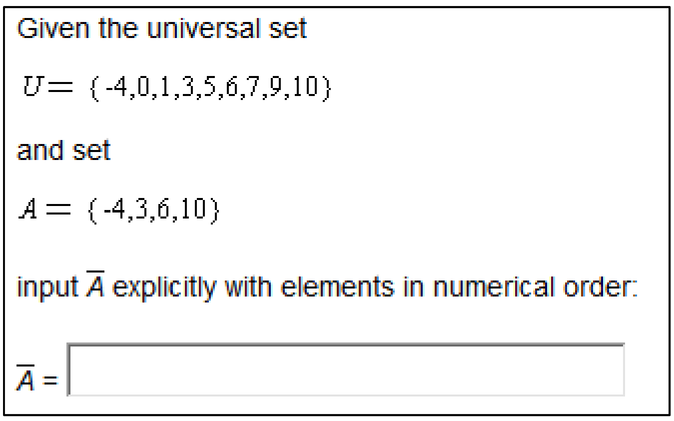


Figure 4.8: Question on the complement of a set using line above it representing the operation.

4.8.1. Assumed and tested skills

It is assumed that students attempting questions on the complement of a set possess some knowledge prior to this. Similarly to the questions on the difference of two sets, they are expected to know the **set-roster notation** (described in Section 4.1.1 and in Section

4.7.1). In addition to these assumed skill, students should know the union and intersection of two sets described in Section 4.6.1.

Students are tested on finding the **complement** of a set when the set (or sets) are given with elements listed explicitly. “The complement of *A*, denoted A^c, is the set of all elements in *U* that are not in *A*” (Epp, 2010, p 341). However, there are other possible symbolic notations of this operation. Rosen (2007) denotes the complement of set *A* with the line above it, i.e. $\bar{A}$, whereas Grassmann (1996) indicates it by the tilde character, i.e. ~*A*. Like in the questions on other operations, here also the process of finding the correct answer involves the **pairwise identification** of elements between sets. This time however, regardless of the question, the members of a set are matched with the elements of a universal set as indicated by the definition. Students are also tested whether they know the optional representations (symbolic and words) of the complement of a set.

4.8.2. Objective questions

Word input type questions

Questions on the complement of a set are, like those on elements, union, intersection, and difference, of the word input type. Therefore, for the description of required formats of answers please see Section 4.1.2 and Section 4.6.2.

Different styles

All questions provide students with the sets given explicitly. Four questions ask for the complement of set *A*, three questions for the complement of $A \cup B$, and four questions for the complement of $A \cap B$. Other parameters differentiating the styles of the questions are of the notation used for the complement. Three questions use an overbar, eight questions use superscript ‘c’, and two describe operation using words (see Figure 4.8.2). Further, restrictions on some sets are applied. On the top of questions where no restrictions on sets are present, sets may be subject to some constraints: it is the universal set, it is an empty set (see Figure 4.8.3), or it is neither the universal nor an empty set.

Let the universal set

$U = \{-2,0,2,5,6,7,8,9,10\}$

and two sets

$A = \{6,7,9,10\}$

$B = \{0,2\}$

Set C comprises elements that are neither in A nor in B. Input C explicitly with elements in numerical order:

C =

Figure 4.8.2: Question on the complement of a union of two sets using words representing the operation.

The reason for differentiating the database by the separation of the special cases is to identify students' strengths and weaknesses. This would not be possible if all these combinations of the questions would be resulted from one question, by random selection.

Random parameters

Random parameters include the size of sets presented to students as well as the members chosen at random. However, since the complement of a set depends on the universal set that is also a subject for the different question styles; the coding involved the number of constraints on them (see Section 4.8.4) allowing separate cases in the questions.

4.8.3. Feedback

The feedback in Figure 4.8.3 contains the definition of the universal set together with the set itself from the question. Then, since the question asks for the complement of the intersection of sets *A* and *B* the definition of the intersection is given together with the set that should be found. Finally, the complement of a set can be found. The explanation of the operation and the solution is provided.

Given the universal set

$U = \{-1,1,4,6,7,8\}$

and two sets

$A = \{4,8\}$

$B = \{1,4,6,7\}$

input $(A \cap B)^C$ explicitly with elements in numerical order:

~~~~~~Your result~~~~~~

Your answer, **{4}**, should have been **{-1,1,6,7,8}**.

The universal set is a set containing all elements of a problem under consideration. In the question, the universal set is given by U = {-1,1,4,6,7,8}.

The intersection of two sets A and B, denoted by A∩B, is the set of elements which belong to both A and B. Since A = {4,8} and B = {1,4,6,7}, A∩B = {4}.

The complement of the intersection of two sets A and B, denoted $(A \cap B)^C$, is the set of elements which belong to U but which do not belong to A∩B.

Therefore, the correct answer for $(A \cap B)^C$ should be {-1,1,6,7,8}

Figure 4.8.3: Feedback on complement of the intersection of two sets using words to represent the operation.

4.8.4. Technical content

Technical content on the complement of a set will be presented in this section. Sets are given in the set-roster notation and the process of creating sets is the same as the one described in Section 4.2.4. Despite of the fact that Figure 4.8.3 gives only sets *A* and *B*, set *C* below was also coded to help to separate the special cases. It was used, for example, to build set *B* so it will not have common elements with the set *A* or to add more elements to the universal set.

- Defining the universal set

universal = set_union(abunion_string,cset_string);
universal_string=universal.join();

set_union function is used to find the universal set. It is formed by two sets: the union of sets *A* and *B* (coded using the *set_union* function on sets *A* and *B*), and set *C*.

- Finding the complement of a set

```
complement = set_complement(abintersection_string,universal_string);
```

set_complement (see Appendix E) is a function in *brunel_logic*. It finds the complement of a set and returns it with elements in an alphabetical order. It requires two parameters: string of elements of the set $A \cap B$ (coded using the *set_intersection* function), and the universal set string.

- Confirming the final look of the universal set and the complement set

```
dset="10";
if (complement==""){universal = set_union(dset,universal_string);}
universal_string=universal.join();
complement = set_complement(abintersection_string,universal_string);
```

One of the constraints on the question is that the intersection of sets *A* and *B* is not the universal set. This happen when its complement is empty and therefore this is checked and a condition included for adding an extra element to the already existing universal set for such cases. 10 is the number to be added as it is never a member of the universal set.

Finally, if the universal set has been amended, the complement set also has to be changed according to the new universal set.

4.9. Partitions

The database comprises four questions on partition of a set. Questions are of word input and numerical input type and of different styles. Each question asks students to determine different aspects of the partition of a set. Two problems give sets explicitly, whereas the other two involve partition formed by the relation R.

An example of a question on the partition of a set is presented in Figure 4.9.

Let

$A = \{-8,-6,-5,-4,3,4,5,6,14,15\}$, $A_1 = \{14,15\}$, $A_2 = \{-6,-5,-4\}$, and $A_3 = \{3,4,5,6\}$

Is $\{A_1,A_2,A_3\}$ a partition of A?

Please answer YES or NO.

Figure 4.9: Question on the partition of a set.

4.9.1. Assumed and tested skills

Whether all the elements of the set are listed, or due to the size of the set only smallest and largest are given, it is assumed that students are familiar with the **set-roster notation** (Section 4.1). Where necessary, it is assumed that students know what the **relation *R*** is as well as the **modulo *n*** (Section 3.2.1). Given two sets *A* and *B*, "a relation R from *A* to *B* is a subset of *A*×*B*. Given an ordered pair (*x*, *y*) in *A*×*B*, *x* is related to *y* by R." (Epp, 2010, p. 14). Modular arithmetic gives a way of generating equivalence relations that can be used to partition sets of integers, as in Figure 4.9.2a.

A number of terms are used in the definition of the partition of a set. Therefore, students are expected to know all of them in order to be tested when they are put together. A partition of a set A is a collection of nonempty **subsets** (defined in Section 4.2.1) such that those subsets are mutually disjoint and A is the **union** (defined in Section 4.6.1) of those sets. **A nonempty set** is a set with at least one element. **Mutually disjoint** sets are sets which do not have any elements in common.

Students are tested on the understanding of the definition of the partition and whether they are able to use it while defining different information on the partition of the set.

4.9.2. Objective questions

Numerical input type questions

One of the questions with partition formed by the relation and involving modulo n is of the numerical input type (Figure 4.9.2a). Students have to state how many subsets are in the given partition. This depends on the value of *n*.

Let R be the relation
$x \mathrm{R} y \Leftrightarrow x \bmod 7 = y$

on the set
$D = \{2,3,4,5,\ldots,21,22\}$

How many sets is in the partition of D?

Figure 4.9.2a: NI type question on partition of a set.

Word input type questions

The remaining three questions are of the word input type. Students have to type a string of characters. Instructions on the required format of the input are given. In one question it is either ‘YES’ or ‘NO’ (Figure 4.9), whereas in the other two the expected answer is a set and therefore descriptive guidelines are presented.

Random parameters

The questions where partitions are formed by relations use a number of random parameters. Letters representing sets are chosen at random, the value of *n* of the modulo is also selected each time the question is run and takes values between 4 and 8. The set for the partition is also selected at random. It is of different cardinality as it is defined by choosing its smallest and largest elements and it includes all numbers between them. One question gives students one element and asks for the whole subset of the partition (Figure 4.9.2b). This element is chosen at random.

Let R be the relation
$x \mathrm{R} y \Leftrightarrow x \bmod 7 = y$

This is used to partition the set
$H = \{-3,-2,-1,0,\ldots,21,22\}$

into a set of subsets of H.

18 is one of the elements of a subset in the partition of H.
State this subset.

Important: input your answer without spaces, separating each element with a comma and in ascending order e.g. {1,2,3}.

Figure 4.9.2b: WI type question on partition of a set.

In the remaining questions randomised the cardinality of all given sets and their elements (e.g. Figure 4.9).

Different styles

As mentioned already and described while explaining different types of questions and random parameters, two problems give the sets explicitly, whereas the other two involve partition formed by the relation R. However, each of the questions asks different questions. One simply asks whether the given sets form a partition or not (Figure 4.9), another asks for the number of subsets in the given partition (Figure 4.9.2a). The other two ask for one of the subsets of a partition, either knowing one of its elements (Figure 4.9.2b) or knowing the remaining subsets (Figure 4.9.3).

4.9.3. Feedback

The screen shot of the feedback screen for the fourth question on partitions and not presented yet is presented in Figure 4.9.3 below. Similarly to the other questions students are firstly provided with the question they were given, their answer and in the cases when it was wrong they are informed about the correct answer and the explanation how it could be obtained. This on the other hand comprises the definition of the partition (also present in the feedback for other questions in this set) and the step by step solution involving operations on sets.

Let

$A = \{-5,-4,-2,-1,0,1,2,4,5,6,7,8\}$, $A_1 = \{-5,-1,0,1\}$, $A_2 = \{4,5,7,8\}$

What is the set A_3 so that $\{A_1,A_2,A_3\}$ is a partition of A?

<u>Important:</u> input your answer without spaces, separating each element with a comma and in ascending order e.g. {1,2,3}.

~~~~~~Your result~~~~~~

Your answer, **{-5,-2,0,1,8}**, should have been **{-4,-2,2,6}**.

A partition of a set A is a collection of nonempty subsets A_1, A_2, A_3, ... such that those subsets are mutually disjoint (no two sets have any elements in common) and A is the union of those sets.

With A, A_1 and A_2 given, a way to find A_3 is to firstly find $A_1 \cup A_2$

$A_1 \cup A_2 = \{-5,-1,0,1\} \cup \{4,5,7,8\} = \{-5,-1,0,1,4,5,7,8\}$.

Then you have to identify elements in A that are not in $A_1 \cup A_2$, i.e.

$A \setminus (A_1 \cup A_2) =$ **{-4,-2,2,6}**.

This is your answer for A_3 since this set does not contain any elements that are either in A_1 or in A_2, and $A_1 \cup A_2 \cup A_3 = A$.

Figure 4.9.3: Feedback of the questions on finding a subset of a partition when remaining subsets are given.

According to the information in Section 4.2.1, whenever one set is a proper subset of another one (e.g. $B \supset A$) then two other relations are true, i.e. $B \supseteq A$ and $B \neq A$.

4.9.4. Technical content

Technical content on listing the elements of a set will be presented in this section. All the coding will be shown on the example presented in Figure 4.9.3 and this is as follows:

- Correct answer

Sets A, A_1, and A_2 are generated in the same way like set B described in Section 4.2.4. Set A_3 on the other hand, is built using *set_union* and *set_difference* functions corresponding to the operations on the feedback screen.

universal_string=unionabc_string;

difference = set_difference(unionabc_string,unionbc_string,universal_string);

universal_string is a list of elements in the universal set and is the same as those in the union of three sets randomly chosen. *difference* represents set A_3 and is obtained using *set_difference* function being a difference between *unionabc_string* (set A) and *unionbc_string* (union of sets A_1 and A_2). At this stage *difference* can be empty in few cases, but since the definition of a partition does not allow it additional lines of coding are necessary.

if(difference == ""){unionabc=set_union(unionabc_string, "-9")}
else{unionabc=unionabc}

In the case when *difference* (set A_3) is empty, '-9' is added to the set A using *set_union* function on the existing set and '-9'.

4.10. Algebra (Set identities)

The examples presented and described so far in the topic on sets were based on the sets with specified elements. This section, however, talks about **set identities** that are universally true for all sets.

The database comprises of three questions. They are of 4TFUSP type that requires four inputs from the selection of three possibilities: T (true), F (false), U (undecidable), and are of different styles.

An example of a question on set identities is presented in Figure 4.10 below.

Let U be the universal set and let A,B and C be any subsets of U with the properties that there is at least one element in each set that is not in the other two sets, and that U contains elements that are not in any of these sets.

Consider the following table of statements.

If you think the statement is True, type T.
If you think the statement is False, type F.
If you think the statement is Undecidable, type U.

Statement	T, F or U ?
$A \cup U = \emptyset$	
$B \cup A = A \cap B$	
$\overline{(A \cap B)} = \overline{A} \cup \overline{B}$	
$A \cup (A \cap B) = A \cup B$	

Remember all inputs must be either T, F or U.
Obviously with this limited number of input choices, a randomly-guessing person could do quite well on this question. To gain a mark, you therefore need to get every input correct

Figure 4.10: Question on set identities.

4.10.1. Assumed and tested skills

It is assumed that students know the definitions and symbolic notation of operations on sets. They include: union, intersection, difference, and complement. Furthermore, they should know the concepts of the universal and empty sets as well as their symbolic notations U and $\emptyset$, respectively.

Students are tested on their knowledge of set identities. Twelve of them can be distinguished and all of them randomly appear in the questions. They are summarised in the Table 4.10.1 below.

Table 4.10.1: Set identities (Epp, 2010, p. 355)

Names of the laws and specifications of sets	Identities
Commutative Laws: For all sets A and B,	$A \cup B = B \cup A$ $A \cap B = B \cap A$
Associative Laws: For all sets A, B, and C,	$(A \cup B) \cup C = A \cup (B \cup C)$ $(A \cap B) \cap C = A \cap (B \cap C)$
Distributive Laws: For all sets A, B, and C,	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
Identity Laws: For all sets A,	$A \cup \emptyset = A$ $A \cap U = A$
Complement Laws:	$A \cup A^c = U$ $A \cap A^c = \emptyset$
Double Complement Law: For all sets A,	$(A^c)^c = A$
Idempotent Laws: For all sets A,	$A \cup A = A$ $A \cap A = A$
Universal Bound Laws: For all sets A,	$A \cup U = U$ $A \cap \emptyset = \emptyset$
De Morgan's Laws: For all sets A and B,	$(A \cup B)^c = A^c \cap B^c$ $(A \cap B)^c = A^c \cup B^c$
Absorption Laws: For al sets A and B,	$A \cup (A \cap B) = A$ $A \cap (A \cup B) = A$
Complements of U and ∅:	$U^c = \emptyset$ $\emptyset^c = U$
Set Difference Law: For all sets A and B,	$A - B = A \cap B^c$ (alternative notation is $A \setminus B = A \cap B^c$)

One question tests students directly on the knowledge of these identities, whereas two others test more complex thinking as they include statements of combinations of the above laws.

4.10.2. Objective questions

4TFUSP input type questions

The questions contain four statements of which correctness has to be defined. They allow inputting one of three possibilities for each statement; however, only true or false can be right. For this to hold, introductory sentences are given at the beginning of each problem. This prevents cases where it would be impossible to give a clear answer and ‘undecidable’ would have to be selected.

Different styles

As mentioned already one question includes a table with set identities laws. The other questions are based on these identities, making more complex identities.

Random parameters

In each of the questions, the set identities are chosen at random. They are formed by the random selection of eight sets. They are chosen from the separate collections of sets used for the left and right sides of the identities.

4.10.3. Feedback

Similar to the feedback of the TFU questions presented in Section 4.2.3, the feedback of the 4TFUSP type questions consists of the question itself followed by a table. It contains submitted answers. If the answers are correct they are displayed in green, if wrong in red. Below, depending on the question, students are given the explanation of each identity they provided the wrong answer.

In the feedback students are given an explanation of the particular law. However, the feedback of more complex questions (Figure 4.10.3) includes the advice to draw a Venn diagram (explained in Section 4.12.1) and the hints of which laws should be used to correctly answer the question.

Let U be the universal set and let A and B be any two distinct non-empty proper subsets of U such that neither is a subset of the other. Moreover $A \cup B \neq U$.

Consider the following table of statements.

If you think the statement is True, type T.
If you think the statement is False, type F.
If you think the statement is Undecidable, type U.

Statement	T, F or U?
$A \cap (\overline{B} \cup \overline{A}) \cup B = A \cup B$	Your input F should have been T
$A \cap ((\overline{\overline{A}}) \cup \overline{B}) \cap B = \emptyset$	Your input T should have been F
$(A \cap B) \cup (A^c \cup B)^c = A$	Your input T is right
$(A \cap B)^c \cup (A^c \cap \emptyset)^c = A \cap B$	Your input F is right

~~~~~~Your result~~~~~~

You got two values right. Input3 = T and input4 = F . However, input1 = T and input2 = F .

$A \cap (\overline{B} \cup \overline{A}) \cup B = A \cup B$
If you have any problem understanding why this is the correct answer, try drawing a Venn diagram for both sides of the equation, or use the following laws: Commutative Law, Distributive Law, Complementary Law and Identity Law.

$A \cap ((\overline{\overline{A}}) \cup \overline{B}) \cap B = A \cap B$
If you have any problem understanding why this is the correct answer, try drawing a Venn diagram for both sides of the equation, or use the following laws: Double Complement Law and Absorption Law.

Figure 4.10.3: Feedback of the question on the set identities.

4.10.4. Technical content

Not only the feedback screen of the questions on subsets and set identities are similar, but also coding. Setting up a matrix of 'F's as well as setting up non false elements is similar to Section 4.2.4.

All coding will be shown for the example presented in Figure 4.10.3 and this is as follows:

- Defining a list of statements and propositions

```
var subjects = new Array(""," (<i>A</i> &cap; <i>B</i>) &cup;
"+Acompl_unionB_complement+" "," (<i>A</i> &cup; <i>B</i>) &cap;
"+AunionB_complement+" ",(…)," <i>B</i> &cap; (<font style='text-
decoration:overline'><i>A</i></font> &cup; <font style='text-
decoration:overline'><i>B</i></font>) &cap; <font style='text-
decoration:overline'><i>A</i></font> ");

var Nrow = subjects.length -1;
var properties = new Array("","<i>A</i>", "&#216;", "<i>U</i>", " <i>A</i> &cup; <i>B</i>
"," <i>A &cap; B</i> ");
var Ncolumn = properties.length - 1;
```

subjects is an array containing *Nrow* (here 16) strings that are possible left hand sides of candidate statements. *properties* is an array containing possible matches for the *subjects* and there are five of them here so *Ncolumn*=5.

- Building up the feedback

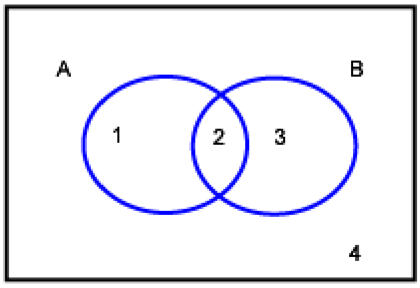
```
feedback = new Array();
(…)
for(i = 1; i < Nrow+1; i++){
feedback[i] = "<p>"+subjects[i]+"= "
for(j = 1; j<= Ncolumn; j++){
(…)
if(SPmatrix[i][j] == "T"){feedback[i] += ""+properties[j]+" <br>"+f+""
TFs = 1;
for(k = Ncolumn; k > j; k--){
if(SPmatrix[i][k] == "T"){TF = 0}else{TF=1}
TFs = TFs*TF;}

if(TFs == 0){feedback[i] += "  "}}}}
```

Firstly an array *feedback* is defined. Then, *subjects* (sets of the left side of identities) are the first parts of the feedback for each of the statements*. T*he coding (not presented here due to its length) includes the explanations of each identity and they are represented by *f* for each of the sixteen *subjects*. Going back to the presented coding, each of the entries of the matrix that has ‘T’ assigned adds to the feedback the *property* (set of the right side of identities) together with the explanation *f*.

4.11. Venn Diagrams

The database comprises two questions on Venn diagrams. Questions within this subtopic are of word input type, but of different styles. The instructions are the same for each of the questions. They differ by the number of sets in the problem.

Given is the following Venn diagram with indicated two sets A and B and four areas 1-4:



Alice is required to shade in the set $\boldsymbol{A \cap B^C}$.

What area(s) does she shade in?

Important: input your answer without spaces, separating each element with a comma and in ascending order e.g. ***1,2,3*** *or type* ***0*** *if you think the set is empty.*

Figure 4.11: Question on Venn diagrams with two sets.

4.11.1. Assumed and tested skills

It is assumed that students know that Venn diagram is a **graphical representation** of sets and as in the case of the questions it is often used to show **relations** between them. The universal set is usually represented by a rectangle, whereas sets by circles or other geometric figures (Rosen, 2007).

Students are expected to know **set operations**, i.e. union, intersection, difference, complement (described in earlier paragraphs) and their **symbolic notations**.

They are tested on finding the areas of the diagram corresponding to the set defined by the combination of sets and operations on them given in the symbolic notation.

4.11.2. Objective questions

Word input type questions

Students are presented with two or three, depending on the question, overlapping circles representing sets located in the rectangle being the universal set. All the regions are numerated. Students are asked to indicate areas corresponding to sets given in a symbolic form. Therefore, an input should be a list of numbers from the diagram. Its format is described underneath the input box, and includes the information that ‘0’ (zero) should be typed if the set is empty.

Random parameters

If in the question numbers are present they are most likely to be randomised. However, the only numbers involved in the questions are those in the diagrams to represent the regions and they stay unchanged every time the question is run. On the other hand, the symbolic notations of sets are changing. They are selected at random from 20 available in the question involving two sets and from 55 when given is the question with three sets.

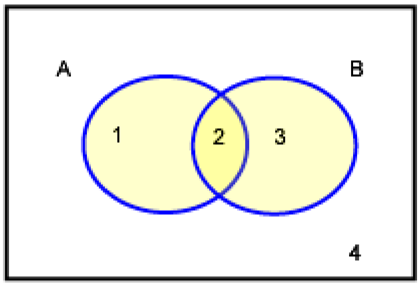
Different styles

The two questions from the database differ by the number of sets involved and, as mentioned above, include either two or three overlapping circles.

4.11.3. Feedback

The screen shot of the feedback screen is presented in Figure 4.11.3 below. It starts with the original question followed by the submitted and correct answers. Since each set in the question is drawn from a group of them and it is impossible to automatically generate explanation of the set operations involved, the reasoning behind the correct answer is not given. However, possible sets matching the regions submitted by students are given. This is possible since available sets are pre-matched with the regions (8 such regions can be distinguished) of the Venn diagram allowing correct marking. At the end students are also asked to try to draw the Venn diagram again for the given set.

Given is the following Venn diagram with indicated two sets A and B and four areas 1-4:



Carmen is required to shade in the set $\boldsymbol{A \setminus B^C}$.

What area(s) does she shade in?

~~~~~~Your result~~~~~~
Your answer, **3**, should have been **2**.

Your answer matches the following set(s) - there are other ways of representing your answer too:

$B \setminus A$

$A^C \cap B$

$A^C \setminus B^C$

Please try to draw the Venn diagram for $A \setminus B^C$ again and hopefully the area(s) **2** will be shaded in. If in doubt, please ask your teacher or fellow students.

Figure 4.11.3: Feedback on Venn diagrams with two sets.

In cases when no matching set can be found to the submitted region students are given the following annotation: 'Either you have been guessing or you made a typing error when putting your answer. Please always double check your answer in the input box.'

4.11.4. Technical content

Technical content on Venn diagrams will be presented in this section. All the coding will be shown on the example presented in Figure 4.11.3 and this is as follows:

- Random selection of the index of an symbolic notation of a set and the correct answer

set_choice = displayarray(1, 1, Nrow, 1);
s0 = set_choice[0];

```
for(j = 1; j<= Ncolumn; j++){
if (SPm[s0][j]=="T"){
Correct%QUESTION.NUMBER% = answers[j];}}
```

sets is an array holding symbolic representations of sets whereas *answers* is an array holding lists of numbers representing areas in the diagram. *s0* is a randomly chosen number indicating which set students will be asked to find on a diagram. *SPm* is a matrix matching sets with areas in the diagram. ‘for loop’ is used to generate the correct answer. It searches through the matrix to find the position j in an *answer* array which then returns one of its elements of the index *j.*

- Matching each of the areas of the diagram to all the available representations of sets in the symbolic notation

```
nothing = "";
for(j = 1; j<= Ncolumn; j++){
eval("Distractor"+j+" = nothing");
for(i = 1; i<= Nrow; i++){
if (SPm[i][j]=="T"){eval("Distractor"+j+" += sets[i] + '<p>' ")}
}}
```

At first *nothing* is defined as an empty string. Later, using the ‘for loop’ on *j* a number of distractors is defined. There is as many distractors as many elements is in the *answers* array (that is *Ncolumn*). Each of them is assigned the *nothing* string. Then, ‘for loop’ on *i* in a matrix assigns sets in symbolic notations to distractors that are related to the possible answers that students could type.

4.12. Counting principle

The database comprises of 18 questions. All of them are of numerical input type, but of different styles. They differ by the number of sets in the problem, but also by the relationship between each other.

A group of 100 students plan to study Mathematics, Programming and Engineering as part of their course. 14 plan to study only Mathematics, 29 only Programming and 37 only Engineering. 10 will study Mathematics and Programming, 9 Mathematics and Engineering, 11 Programming and Engineering. How many plan to study 2 of 3 subjects, given that each student must study at least one of the 3 subjects?

Figure 4.12: Question on counting principle.

4.12.1. Assumed and tested skills

It is assumed that students can assimilate all the information that is presented to them by the question.

They are tested on understanding the English translation of the sets. In the presented example, students are examined if they know that given that '29 students will study only programming' is the set of programming taking away elements common with engineering and mathematics; '9 students study mathematics and engineering' is the intersection of the two sets; 'number of students studying 2 of 3 subjects' however, is the union of three subsets: intersections of each pair of sets but without the set belonging to each set.

After students translated each of the statements, they are tested on the counting principle: $n(A \cup B \cup C) = n(A) + n(B) + n(C) - n(A \cap B) - n(A \cap C) - n(B \cap C) + n(A \cap B \cap C)$, that involves number of elements in each of the listed sets; or on usage of Venn diagrams, by pictured representation of sets as regions, to find the answer. There is no preferred method that takes students to right answer, so both are given.

4.12.2. Objective questions

Numerical input type questions

Questions provide students with the rich information of sets and the number of elements either in each set or in more than one set. The question is dressed in the subject of university subjects. Therefore, instead of describing elements of sets *A*, *B*, and *C* for example, students are given the numbers representing the number of students studying mathematics, programming, and engineering. Their task is to find a number representing the number of students allocated to a specified subset.

Different styles

As mentioned already questions differ by the number of sets involved in the problem, and there are either 2 or 3 sets. The other difference is the set whose number of elements students are asked to find. Lastly, they may differ by the possibility of elements that do not belong to any set except the universal set.

Random parameters

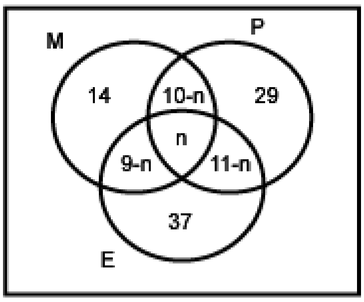
All the numbers involved in the questions are chosen at random.

4.12.3. Feedback

The feedback gives two ways of solving the question. The first method is by using Venn diagram. It contains all the numbers that were present in the question followed by the step-by-step calculations leading to the correct answer. From there the counting formula that is presented to students follows.

~~~~~~Your result~~~~~~
Your answer 3, should have been 28

One of the ways to solve this problem is to draw Venn diagram. Let M, P and E be the number of students studying Mathematics, Programming and Engineering respectively. Then we denote the number of students taking all 3 subjects by *n* and hence we are able to illustrate all the given information by the following Venn diagram.



There are 100 students in the group and each of them must study at least one of the 3 subjects.
Thus, from the Venn diagram,

$14 + (10 - n) + n + 29 + (11 - n) + 37 + (9 - n) = 100$
$110 - 2n = 100$
$n = 5$

indicating that 5 students will study all 3 subjects.

The number of students planning to study Mathematics is the sum of elements in three regions:

$14 + (10 - n) + n + (9 - n) = 14 + (10 - 5) + 5 + (9 - 5) = 28$.

<u>Using the counting formula</u>
Drawing a Venn diagram is useful: you will then be able to understand the general formula for counting elements in sets i.e.

$n(M \cup P \cup E) = n(M) + n(P) + n(E) - n(M \cap P) - n(M \cap E) - n(P \cap E) + n(M \cap P \cap E)$

Now you can input the numbers you know into this formula and rearrange to give the unknown number. Then you know the numbers of elements in each part of the Venn diagram. *Read the question carefully to see what is asked for!*

Figure 4.12.3 Feedback on counting principle.

4.12.4. Technical content

The technical content in this section will be presented for the example presented in Figure 4.12.3. Most of it is the same for all the questions and differs only by some wording and the correct answer.

- Allocation of the number of elements for each set

```
myAArray = displayarray( 1, 2, 15, 0);
a = myAArray[0];
myCArray = displayarray( 1, 2, 7, 0);
c = myCArray[0];
myKArray = displayarray( 3, 1, 7, 0);
mp = myKArray[0];
pe = myKArray[1];
me = myKArray[2];
cmp = c + mp;
cpe = c + pe;
cme = c + me
myPArray = displayarray( 1, 15, 35 , 0);
p = myPArray[0];
myEArray = displayarray( 1, 15, 99-a-p-cpe-me-mp , 0);
e = myEArray[0];
m = 100-a-p-e-pe-me-mp-c;
g = 100-a;
r = m+p+e+cmp+cpe+cme;
```

a represents the number of elements not belonging to any of the sets; *c* those in the intersection of all three sets; *mp, pe, me* number of elements of intersections of two sets but without *c* (the elements of intersection of all three sets. All these parameters were chosen at random. Now they are used to build up the intersections of two sets including the intersection of all three sets, simply by adding *c* to *mp, pe,* and *me,* forming *cmp, cpe,* and *cme.* Also at random are chosen the numbers of elements unique to sets without common elements, resulting in *p* programming), *e* (engineering), *m* (mathematics) being defined. *g* is the union of all three sets. *r* is a union of sixt sets required for the calculation.

- Defining entries of the Venn diagram and generating it

```
numbers = new Array();
numbers[0]=m;
numbers[1]=cmp+"-n";
numbers[2]=p;
numbers[3]=cme+"-n";
numbers[4]="n";
numbers[5]=cpe+"-n";
```

```
numbers[6]=e;
numbers[7]=a;
```
svg_Venn= SVG_Venn(labels,numbers);

numbers is an array containing all the entries that will be allocated in the Venn diagram. When three sets are overlapping, seven regions are distinguished. *SVG_Venn* is a function in *brunel_svg* and requires two parameters: labels of sets and entries filling in each area of a diagram.

- Correct answer

Correct%QUESTION.NUMBER% = mp+pe+me;

The set of which number of elements students have to find is pre-decided, so no calculation is needed. The question is reverse-engineered from these random parameters.

5. Logic

> "Logical reasoning is the essence of the mathematics and, therefore, it is an important starting point for study of discrete mathematics. Logic is concerned with methods of reasoning. One of the main aims of logic is to provide rules by which we can determine whether any particular argument is valid or not… Any collection of rules or any theory needs language in which these rules or theory can be expressed. Natural languages are not always precise enough. They are also ambiguous. It is, therefore, necessary to develop a formal language called object language. In order to avoid ambiguity, we use symbols which have been clearly defined in the object languages. An additional reason to use symbols is that they are easy to write and manipulate. Because of the use of symbols, the logic that we shall study is known as symbolic logic." (Biswal, 2009, p. 1)

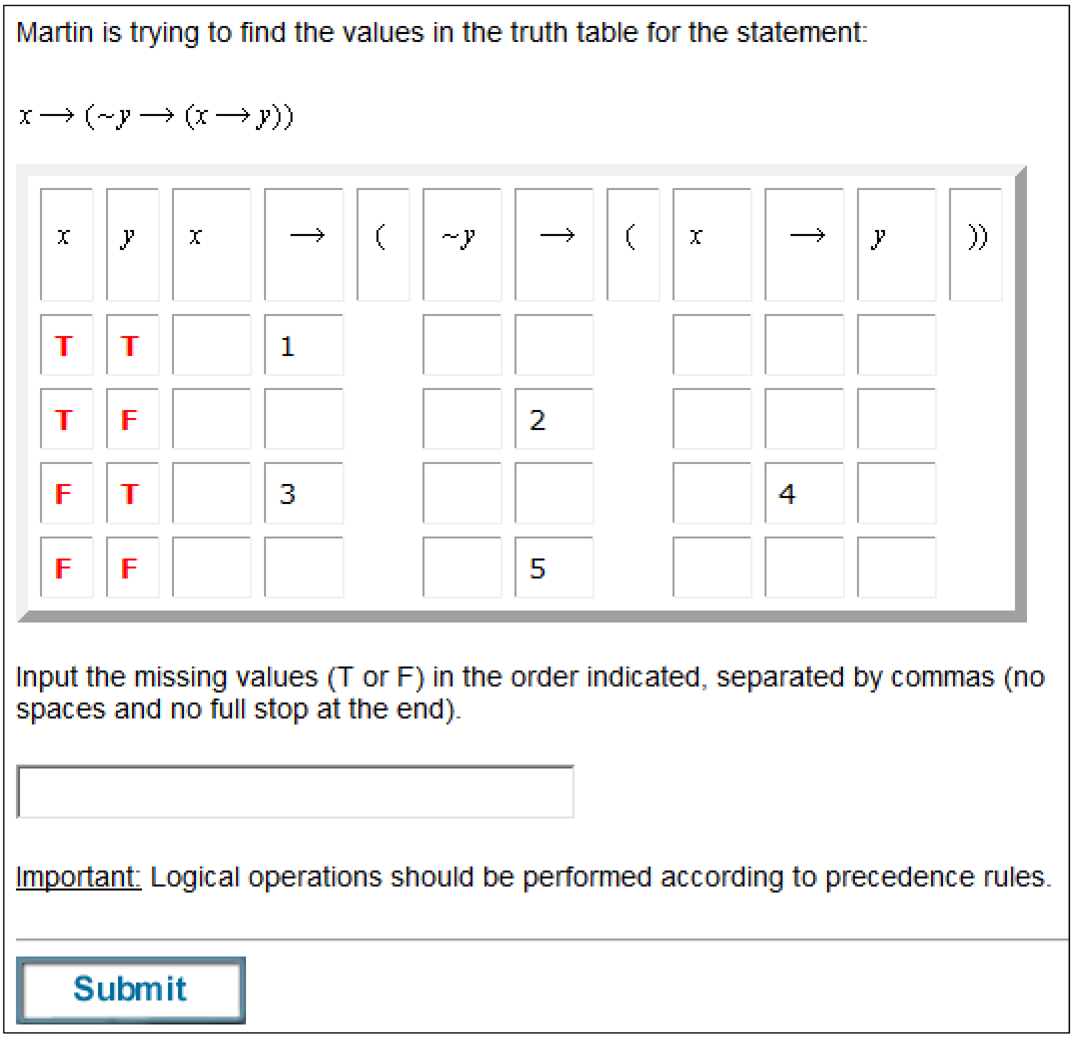
When talking about object languages it is assumed that this is the collection of simple, declarative sentences that can take one of the two possible truth values, true (T) or false (F). These sentences are called *statements* (or *propositions*) and are usually denoted by small letters *p*, *q*, *r*, etc. Furthermore, by using logical connectives such as *not*, *and*, *or*, simple statements can be combined to form *compound statements* for which truth values can also be defined. Each of these connectives has symbols associated to them. Moreover, logical connectives are also called logical operators or sometimes unary or binary connectives whose values are found by the Boolean algebra of truth values. The ability to identify symbols for these connectives and to assign their truth values is just a part of the skills students should possess.

5.1. Truth tables

The database comprises of 51 questions on truth tables. As the name suggests, the questions are on finding the truth values of the propositions presented in the tables. Questions are of the word, i.e. string, input type, but of a different style. They differ from each other by the number of statements, the type of operators, the complexity, and the number of entries students are asked to find.

The word input type is well suitable for this kind of question, because the number of possible inputs is very limited (only F and T are possible), and the expected length of the input string is known to the system. Therefore wrong inputs can be detected easily in many cases. Multiple choice type questions, on the other hand, would not be suitable here. In the context of truth tables, it would be very hard to identify common mistakes made by students and provide these as alternative answers to distract students, which is the purpose of multiple choice type questions. Also as mentioned in Section 2.3 multiple choice questions tend to scaffold a students' thinking.

An example of a truth table question is presented in Figure 5.1 below.


Martin is trying to find the values in the truth table for the statement:

$x \rightarrow (\sim y \rightarrow (x \rightarrow y))$

x	y	x	$\rightarrow$	(	$\sim y$	$\rightarrow$	(	x	$\rightarrow$	y	))
T	T		1								
T	F					2					
F	T		3						4		
F	F					5					

Input the missing values (T or F) in the order indicated, separated by commas (no spaces and no full stop at the end).

Important: Logical operations should be performed according to precedence rules.

Submit

Figure 5.1: Question on truth tables, with three operators of the same type, with parentheses.

5.1.1. Assumed and tested skills

Within questions on truth tables it is not possible to distinguish general assumed and general tested skills across the full range of possible truth tables. They vary in their degree of complexity and therefore more complicated propositions will require more knowledge. Therefore, the skills tested on easy examples will become assumed skills for the more complicated examples.

Regardless of the difficulty of the proposition, students should be able to **read a truth table**, i.e. to identify the statements involved, indicate the compound statement for which the truth table is built, understand the size of it (i.e. the number of rows containing all possible combinations of truth values the statements can take) and to understand what the truth values under the compound proposition mean.

Going one step further, students are expected to know how to **produce the truth table** themselves. There are four sections in a truth table (see Table 5.1.1a) and students are expected to know how to fill them in.

Table 5.1.1a: Format of the truth tables.

Section 1	Section 2
Section 3	Section 4

Firstly, the compound statement for which truth values have to be found should be written down in section 2. Then, its propositions (p, q, r etc.) should be identified and listed in a single row of section 1. After that students should show ability to lay out a truth table in a logical way in section 3. What is meant by this is that they should use a pattern when filling in rows with truth values below the statements listed in section 1. The standard method is as follows: the columns under the letter representing the first statement should be divided in half (i.e. 2^1 parts for the first statement), the top part should contain T's whereas the bottom F's; the area under the next letter should be divided into 2^2 parts, each part should be filled in alternately with T's and F's starting with T's. Depending on the number of statements the area under the n^{th} statement would be divided into 2^n parts and each part should have T's and F's listed alternately giving 2^n rows. Lastly, it is expected from students to know that section 4 should be filled in with truth values for the proposition written in section 2. This is the only section that is actually tested by the question developed here. However, students are expected to learn how to fill in sections 1-3 from the many examples they are exposed to – see Chapter 10 for further discussion.

Asking students to find **truth values** of propositions with one operator allows clear specification of the tested skill. Three elementary operations can be distinguished: negation, conjunction, disjunction, as well as few more complicated operations that can be built up from these three (e.g. implication, $p \to q \equiv \sim p \vee q$ or equivalent). All of them form a set of eight connectives that are used within the questions and are as follows:

1. Negation – if *p* is a statement, the negation of *p* is 'not *p*' denoted ~*p*. This means that if *p* is true, ~*p* is false; if *p* is false, ~*p* is true. It is a unary operator since it acts on just one statement (in contrast to all the others listed below).

2. Conjunction – if *p* and *q* are statements, the conjunction of *p* and *q* is '*p and q*', denoted $p \wedge q$. It is true only when both are true and false in other cases.

3. Disjunction – if *p* and *q* are statements, the disjunction of *p* and *q* is '*p or q*', denoted $p \vee q$. It is false only when both are false and true in other cases.

4. Implication (conditional statement) – if *p* and *q* are statements, the conditional of *p* and *q* is '*if p then q*', denoted $p \rightarrow q$. It is false only when first statement is true and the second is false, true in other cases. It is best thought of as a 'broken premise'.

5. Equality (biconditional statement) – if *p* and *q* are statements, the biconditional of *p* and *q* is '*p if and only if q*', denoted $p \leftrightarrow q$. It is true only when both statements have the same truth value and false in other cases.

6. Exclusive disjunction (XOR) – if *p* and *q* are statements, the exclusive disjunction of *p* and *q* is denoted $p \oplus q$. It is true both statements have different truth values, false in other cases.

7. Logical NAND – denoted $p \mid q$. It is false when both *p* and *q* are true, true in other cases.

8. Logical NOR – denoted $p \downarrow q$. It is true when both *p* and *q* are false, false in other cases.

All of the above operations have truth values assigned. They are defined in words above, but all the possible combinations of truth values can also be presented in the table called truth table (Table 5.1.1b).

Table 5.1.1b: Truth table containing logical operators.

p	q	$\sim p$	$\sim q$	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$	$p \oplus q$	$p \mid q$	$p \downarrow q$
T	T	F	F	T	T	T	T	F	F	F
T	F	F	T	F	T	F	F	T	T	F
F	T	T	F	F	T	T	F	T	T	F
F	F	T	T	F	F	T	T	F	T	T

Generally, one could be asked for the equivalent representation of statements with operations 4-8 above using $\sim$, $\wedge$ and $\vee$. However, all the statements can also be constructed using only NAND or only NOR (Epp, 2010, pp. 74-75). For example, the NOT, AND, and OR connectives can be written using NAND alone in the following way:

$\sim p \equiv p \mid p$,

$p \wedge q \equiv (p \mid q) \mid (p \mid q)$,

$p \vee q \equiv (p \mid p) \mid (q \mid q)$,

or using NOR alone:

$\sim p \equiv p \downarrow p$,

$p \wedge q \equiv (p \downarrow p) \downarrow (q \downarrow q)$,

$p \vee q \equiv (p \downarrow q) \downarrow (p \downarrow q)$.

The three remaining operations could be expressed using NAND or NOR only by using their equivalent forms, e.g.

$$p \to q \equiv \sim p \vee q,$$

$$p \leftrightarrow q \equiv (p \wedge q) \vee \sim (p \vee q),$$

$$p \oplus q \equiv (p \vee q) \wedge \sim (p \wedge q),$$

and the results for $\sim$, $\wedge$ and $\vee$. These relations are not tested here, but are recommended in Section 11.5 for future work.

For more complicated propositions, knowledge of connectives' truth values is now an assumed skill. As for mathematical operations, logical operations must also be performed according to **precedence rules**. Therefore, this is another assumed skill. However, different sources give different order of precedence. According to Epp (2010), $\sim$ should be performed first, then operations $\wedge$ and $\vee$ as they are considered to be coequal, as are $\to$ and $\leftrightarrow$ that should be evaluated third. On the other hand, some authors, including Rosen (2007), state that operations should be executed in the following order: $\sim, \wedge, \vee, \to, \leftrightarrow$. However, Rosen also suggests using parentheses for clarity. To avoid ambiguity, parentheses are used throughout when conjunction and disjunction, or conditional and biconditional statements are both present. Consequently all the questions have been formulated in the way that students that were taught either of two rules should be able to solve them.

5.1.2. Objective questions

Word input type questions

Because the possibility of constructing varying questions is limited (see Section 5.1), all of them are of the word input type. Figure 5.1 shows an example of students being asked to type their answers in the provided input box. The submitted answer should be in an appropriate format: a string of the mixture of three characters: ‘T’, ‘F’ and comma. Letter T indicates true value for operation, F stands for false value and commas are used to separate T’s and F’s. Participants are also instructed that there should be no spaces between and no full stop at the end of the answer. Hence, answers should be of *T,T,F,T,F* format. If they input spaces the marking function removes them; this also removes trailing spaces which are not even visible to the students.

Different styles

The database of the truth tables’ questions contains questions that are clones of each other. Every question is of different style as compound propositions were obtained by changing the number of statements and operators.

The number of statements used for each problem varies between two and three. The number of propositions specifies the size of the truth table. Number of rows in the table depends on the number of combinations of values that propositions can have and is equal to 2^n, where *n* is the number of propositions. Therefore, tables with two variables will have 4 rows, with three variables 8 rows, whereas with four variables 16 rows. 16 rows makes the table big and difficult to read (especially for CAA where one would need to scroll up and down), so, although quite possible, no questions have been created that will result in this size. It could also be a problem when displaying the table on the computer screen when a big font size is being used. In any case, it is arguable that such questions would not test any new mathematical skill that could not be tested with 3 propositions.

When creating questions on truth tables, eight operators have been used and pre-allocated to each problem. Questions of different styles were created by changing operators and the location of parentheses or their omission. In the situation when questions contain one type of operator, it is clear what knowledge has been tested, whether the student assimilated the tested knowledge, or if more time should be spent to understand specific areas. Formulating problems with mixture of connectives makes questions more interesting and harder to solve. On the other hand, inferring which skills are missing when student gets such a question wrong is more problematic. For instance, getting wrong truth values of statements like $a \rightarrow (\sim b \wedge \sim a)$ (see Figure 5.1.2a or Figure 5.1.2b) may be caused by not knowing one or both of the following: truth values for one or more operations used in the question, i.e. negation, implication, conjunction, disjunction; or precedence rules together with importance of brackets.

In this instance, questions were grouped with respect to the number and type of operators involved in the problem. This allows the test setter to decide what skills will be examined. Questions were classified in the following way:

- Two or three operators of the same type ($\wedge, \vee, \rightarrow$ or $\leftrightarrow$, coupled with negation), with parentheses to avoid ambiguity (Figure 5.1). These questions test students understanding of truth values for specific connectives;

- Two operators of different types ($\wedge, \vee, \rightarrow$ or $\leftrightarrow$, coupled with negation). Symbols $\wedge$ and $\vee$ as well as $\rightarrow$ and $\leftrightarrow$ do not appear together. No parentheses are used (Figure 5.1.3). These questions test knowledge about rules of precedence and of course about truth values;

- Two, three or four operators of different type ($\wedge, \vee, \rightarrow$ or $\leftrightarrow$, coupled with negation), with parentheses (Figure 5.1.2a and Figure 5.1.2b). These questions are more interesting and harder to solve, so require more thinking, more time and more knowledge (e.g. importance of brackets). The more complex the expression is, the more important it is to establish the order in which the table should be built up;

- Three to five operators of different type, including XOR, NAND and NOR, with parentheses. These questions test students' knowledge of $\oplus$, $|$ and $\downarrow$.

Random parameters

Within questions on truth tables a few parameters are set randomly. Firstly, names (e.g. Martin, Claudia, Tania) with which the question starts and proposition names (e.g. q, r, s) are selected at random from the set of all possibilities. These trivial changes are known as 'surface effects' since they do not alter the nature of the questions in any mathematical way. Secondly, missing values from 1 to 5 (or any other number specified by the question designer) are randomly placed in the table. At this point however, possible entries are restricted (see 'Restrictions on possible rows and columns for missing values' in Section 5.1.4 for coding) to make sure that there will be no situation when student is asked for values of direct statements (i.e. q, r, s) or their negations (i.e. $\sim q, \sim r, \sim s$). Therefore, one can be certain that knowledge is tested rather than just the ability to rewrite. Furthermore, the number of missing values can be easily changed (see ‘Number of truth values to be found’ in Section 5.1.4 for coding), and different restrictions can be set up.

An illustration of parameters that change every time the question is run is presented in Figure 5.1.2a and Figure 5.1.2b below.

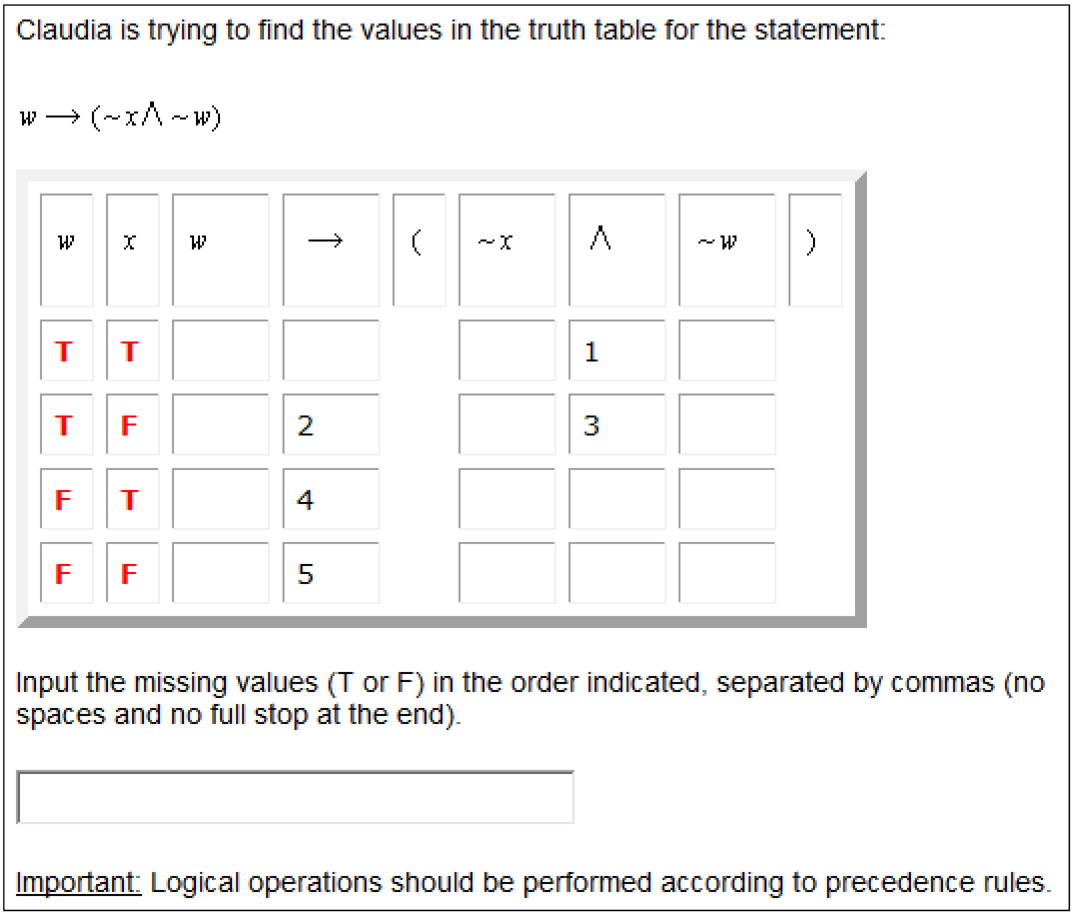


Figure 5.1.2a: Realisation 1 of the truth table question (two operators of different type, with parentheses).

Tania is trying to find the values in the truth table for the statement:

$q \longrightarrow (\sim r \wedge \sim q)$

q	r	q	$\longrightarrow$	(	$\sim r$	$\wedge$	$\sim q$	)
T	T		1					
T	F							
F	T		2			3		
F	F		4			5		

Input the missing values (T or F) in the order indicated, separated by commas (no spaces and no full stop at the end).

Important: Logical operations should be performed according to precedence rules.

Figure 5.1.2b: Realisation 2 of the truth table question (two operators of different type, with parentheses).

5.1.3. Feedback

Questions in the database are formative and summative (see Section 1.1) at the same time. They test the knowledge, but also provide students with detailed and automatic feedback if the submitted answer is wrong. An example of such a feedback screen with description is presented in Figure 5.1.3 below.

~~~~~~Your result~~~~~~
Your answer T,T,F,F,T, should have been T,F,F,F,T

Using the rules of precedence, the truth table for this statement can be built up in the order shown below:

w	x	w	$\leftrightarrow$	x	$\vee$	$\sim w$
T	T	T	T	T	T	F
T	F	T	F	F	F	F
F	T	F	F	T	T	T
F	F	F	F	F	T	T
#		1	3	1	2	1

Figure 5.1.3: Feedback of the questions on truth tables (two operators of different types, without parentheses).

As in all the questions, here also students are presented with the question itself, the submitted response and correct answer, as well as the related material button at the bottom of the screen. The part that is specific to this question is the completed truth table. Together with the truth values for the given proposition, students are provided with the order that the table must be built up indicated in red colour. Combining all this information together should allow students to identify their mistakes and understand what the right answer is. The effect of this information on students' learning will be presented in Chapter 10.

5.1.4. Technical content

Technical content on truth tables will be presented in this section for the selection of questions shown earlier.

All the coding will be shown on the example presented in Figure 5.1 and this is as follows:

- Number of truth values to be found

number_wanted = 5;

Number of entries participant is asked to find is specified in a direct way but can be easily changed as this decision depends on the question designer.

- Number of statements and attributed at random letters

```
terms = new Array;
i = Math.ceil(7*Math.random());
terms[1] = varname_logic(i,0);
terms[2] = varname_logic(i+1,0);
Nvar = terms.length-1;
```

varname_logic is a function in *brunel_logic* template (see Appendix E) that assigns consecutive letters from the string of eleven characters ‘pqrstuvwxyz’ as propositions. For example, if ‘i’ gives value ‘q’ in our example then second proposition is ‘r’.

Nvar within coding stands for a number of propositional variables (*Nvar* = 2 for the example in Figure 5.1). Since every array starts with the 0 index, and it has been left empty, 1 has to be subtracted in order to get the correct number of statements. Here, *terms[1]* represents the first statement and *terms[2]* the second one.

- Size of the table

```
Nrow = Math.pow(2,Nvar);
Ncolumn = Nvar+10;
```

To find the number of needed rows (in addition to headers’ row) *Math.pow(a,b)* JavaScript function has been used. This method returns the value of the first parameter (*2* in our example) to the power of the second parameter (*Nvar* above).

The number of columns is specified by the person creating the question and depends on the compound statement of which truth table has to be found (*Nvar* was 2, after adding 10 the table has 12 columns).

- Restrictions on possible rows and columns for missing values

```
allowed_rows = new Array(1,2,3,4);
allowed_columns = new Array(4,7,10);
```

Missing values of the table are located in every row, but only in selected columns. Columns that are allowed for questioning are those with operators rather than with single or negated statements. This is easily done by direct listing numbers representing rows or columns.

- Assigning truth values of the proposition in the table

```
elements= gettruthtable(MathML_terms,Nvar,Ncolumn);
display_elements= gettruthtable(MathML_terms,Nvar,Ncolumn);
elements[i][Nvar+1] = elements[i][1];
elements[i][Nvar+4] = NOT(elements[i][2]);
elements[i][Nvar+7] = elements[i][1];
elements[i][Nvar+9] = elements[i][2];
elements[i][Nvar+8] = IF(elements[i][Nvar+7],elements[i][Nvar+9]);
elements[i][Nvar+5] = IF(elements[i][Nvar+4],elements[i][Nvar+8]);
elements[i][Nvar+2] = IF(elements[i][Nvar+1],elements[i][Nvar+5]);
```

Function *gettruthtable(MathML_terms,Nvar,Ncolumn)* is stored in the *brunel_logic* template and is given in Appendix E. It returns matrix elements for truth tables that are 0's and 1's. The function sets up first *Nvar* columns in a standard way for every question. *MathML_terms* is an array containing the table headers. Functions 'NOT' and 'IF' are also defined in the *brunel_logic* template and are responsible for assigning the truth values for operators. This allows for coding in a naturalistic and understandable way.

Referring to Figure 5.1, the entry for the missing value indicated by '2' is in Nvar+5=7th column and is obtained by IF operation on the statements from columns 6th (Nvar+4) and 10th (Nvar+8).

- Conversion of 0's and 1's to F's and T's respectively

```
for (i = 1; i <= Nrow; i++ )  {
for (j = 1; j <= Ncolumn; j++ )  {
elements[i][j]=TF(elements[i][j],undecided_string);
}}
```

This is a loop that starts at the 1st row, increases by 1 each time and continue to run as long as the Nrowth row. The same happens with columns and therefore all the elements of the table are converted into T's and F's because of the *TF (a,undecided_string)* function from the *brunel_logic* template.

5.2. Translations

The database comprises of eight questions on translation of symbolic logic into English statements and vice versa. Questions within this subtopic are multiple choice, but as always of different styles. They differ from each other by the instructions, the number of statements and the type of operators (different connectives and many English translations of symbols).

English statements are sentences of subject matter from the ‘Shrek’ movie (*Shrek*, 2001), in order to make questions more interesting and engaging.

5.2.1. Assumed and tested skills

Within questions on translations it is assumed that students know **basic English vocabulary** that enables them to understand the meaning of the sentence. It is further assumed that students understand the rules of **English grammar**. In particular, it is important to be able to distinguish tenses and conditional statements. For example, the sentence ‘If I am carrying an umbrella or it will not rain, I will not get wet’ consists of three simple sentences, namely, ‘I am carrying an umbrella’ contains a subject, a verb in the present continuous tense and an object; whereas both ‘it will not rain’ and ‘I will not get wet’ contain the subject ‘I’, verbs in the future tense and no objects.

Translating an English language sentence into a logical statement is a process of abstraction and the use of replacement rules. Students are tested on three parts of this process. Firstly, students have to be able to understand the **meaning of punctuation, especially the comma**. Inserting commas into the sentence makes the meaning of the sentence clear. For instance, the sentence ‘If I do not revise today’s lecture then I will not be able to understand the future topics and I will fail the exam’ has different meaning depending on whether the comma is either placed before ‘then’ or before ‘and’. Symbolically we have $\sim r \to (\sim u \wedge f)$ versus $(\sim r \to \sim u) \wedge f$. Despite the common-sense meaning of the sentence, in the second version the exam is failed regardless of whether revision takes place or not. When the compound sentence contains three or more simple sentences, commas help to identify where the brackets should be inserted in the symbolic translation of the English statement. The importance of parenthesis has been explained in Section 5.1.1 so connection between commas and brackets is visible.

Secondly, students are tested on their ability to understand the **logical meaning of English words** such as: not, and, or, unless, if/then, since, etc. and their symbolic representatives. Some of these are presented in Section 5.1.1, but the wider selection of most common translations as given in Epp (2010), Rosen (2007), Ikenaga (2009) and Halpin (2006) is shown in Table 5.2.1.

Table 5.2.1: Possible translations of logical operators into English.

Symbolic Form	Translation into English
$\sim p$	Not p It is not the case that p
$p \wedge q$	p and q p but q p as well as q p moreover q p however q p although q p even though q
$p \vee q$	p or q
$p \rightarrow q$	If p, then q If p, q q follows from p p implies q q if p Whenever p, q p is sufficient for q p is a sufficient condition for q a sufficient condition for q is p p only if q q whenever p q when p q is necessary for p q is a necessary condition for p a necessary condition for p is q provided (that) p, q q provided (that) p in case p, q q in case p

$p \leftrightarrow q$	p if and only if q p iff q p is necessary and sufficient for q if p then q, and conversely p is equivalent to q
$\sim p \rightarrow q$	Unless p, then q Unless p, q If not p, then q

The language here conveys additional meaning beyond logic. For example, ‘p but q’ implies ‘p and q’ are both true, but given the truth of p one would not expect q.

Bearing in mind the first two skills, the third tested skill can be defined, i.e., the ability to **parse English sentences into symbolic propositions**. Analysing the above sentence and setting r = ‘it will rain’, u = ‘I am carrying an umbrella’, w = ‘I will get wet’, translating English conjunctions into symbolic connectives, as well as remembering the link between commas and parenthesis, the symbolic form of the English statement ‘If I am carrying an umbrella or it will not rain, I will not get wet’ would be $(u \vee \sim r) \rightarrow \sim w$.

For a translation of symbolic statements to English, similar rules to those presented above apply, but in reverse order. Here, students are tested on understanding the meaning of brackets as the separators giving the sense to the symbolic propositions. They should know that, when translating logical statements into English sentences, commas should be used in the corresponding places to give them the correct meaning. The next skill to be examined is the ability of translating symbolic connectives into English. From combining these two skills, the third tested skill follows. Given the statement in the symbolic form, learners have to identify the corresponding English phrase. This is achieved by using all skills jointly, i.e., the positioning of commas in the right places, correct translation of symbolic connectives and replacement of clauses in the correct English tense for statements to make the sentence readable. Therefore, given the symbolic statement $(\sim m \wedge b) \rightarrow l$ and propositions m = ‘I have money’, b = ‘I go to the bank’, and l = ‘I take a loan’ it should be translated into ‘If I don’t have money and I go to the bank, I’ll take a loan’.

5.2.2. Objective questions

Multiple choice type questions

Students are asked to either translate statements in English into symbolic form (Figure 5.2.2a) or from symbolic into English form (Figure 5.2.2b). Participants have to choose one of the possible six options: four symbolic/English forms (these can be one correct answer and three distractors, or four distractors), 'None of these!' or 'I don't know!'.

If *p* stands for the proposition **Shrek was an ogre** and *q* stands for the proposition **Shrek lived in a swamp** then

Shrek was an ogre only if Shrek lived in a swamp.

is equivalent to:

- ○ $p \wedge q$
- ○ $p \leftarrow q$
- ○ $p \leftrightarrow q$
- ○ $p \rightarrow q$
- ○ None of these!
- ○ I don't know!

Figure 5.2.2a: Translation question with two statements – English into symbols.

If *p* stands for the proposition **Fiona was under a spell** and *q* stands for the proposition **the castle was guarded by a dragon** then the symbolic form

$p \vee q$

could be translated as:

- ○ Fiona was under a spell or the castle was guarded by a dragon.
- ○ Fiona was under a spell and the castle was guarded by a dragon.
- ○ Fiona was under a spell implies the castle was guarded by a dragon.
- ○ Fiona was under a spell is implied by the castle was guarded by a dragon.
- ○ None of these!
- ○ I don't know!

Figure 5.2.2b: Translation question with two statements – symbols into English.

Questions on the translation of statements could have been of the word input type as well. However, this option has been rejected. The reason for this is that questions of this type are checked by the system using string comparison. This would be very tightly specified so that even minor typos would lead to students' answers being marked as false by the system.

Different styles

As mentioned already, questions on translation can be categorised according to the given guidelines: to translate English sentences into symbols, or symbols into English. Both cases have already been presented in Figure 5.2.2a and Figure 5.2.2b.

A challenging aspect of these questions may be that students have to demonstrate understanding of each logical operator, their equivalents in English and finally use of logic in a more contextual way. Also the difficulty level is higher for questions with three statements (e.g. Figure 5.2.2c below) in comparison to those with two statements only (Figure 5.2.2a and Figure 5.2.2b).

If

- *p* stands for the proposition **the castle was guarded by a dragon**
- *q* stands for the proposition **Fiona was under a spell** and
- *r* stands for the proposition **Shrek was an ogre**

then

The castle was guarded by a dragon and ' Fiona was under a spell if and only if Shrek was an ogre '.

is equivalent to:

- $p \leftarrow (q \rightarrow r)$
- $p \rightarrow (q \vee r)$
- $p \wedge (q \leftrightarrow r)$
- $p \vee (q \leftarrow r)$
- None of these!
- I don't know!

Figure 5.2.2c: Translation question with three statements.

In problems with three propositions, two connectives were used out of the possible five: $\wedge, \vee, \rightarrow, \leftarrow$ and $\leftrightarrow$. Since operators ($\wedge$ and $\vee$) and ($\rightarrow$ and $\leftrightarrow$) are considered to be coequal in precedence, parentheses have been used within symbolic logic and commas within English statements to avoid ambiguity. Some of these connectives can be translated into English in more than one way. Possible translations are shown in Table 5.2.1. A selection of them is allows question variation and the testing of broader knowledge.

5.2.3. Feedback

An example of a feedback screen on translation from English into symbols is presented in Figure 5.2.3.

If

- *p* stands for the proposition **Shrek was an ogre**
- *q* stands for the proposition **Donkey was talking a lot** and
- *r* stands for the proposition **Fiona was under a spell**

then

Shrek was an ogre is implied by Donkey was talking a lot, implies and is implied by Fiona was under a spell.

is equivalent to:

~~~~~~Your result~~~~~~

2). Your answer $(p \leftrightarrow q) \leftarrow r$ should have been $(p \leftarrow q) \leftrightarrow r$.

Substituting

- *p* for *Shrek was an ogre,*
- *q* for *Donkey was talking a lot,*
- *r* for *Fiona was under a spell,*
- $\leftarrow$ for *is implied by ,*
- $\leftrightarrow$ for *implies and is implied by*

gives the correct symbolic form: $(p \leftarrow q) \leftrightarrow r$

Figure 5.2.3: Feedback of question on translation from English into symbols.

A distinctive feature of this collection of questions is the information about the substitutions that should be made, and that are necessary to score marks for the correct answer. Students are provided with the English representation for symbolic propositions as stated in the question, but are also reminded about the translation of connectives that were used. Putting together all the conversions should result in the correct translation from words into symbols, or vice versa, depending on the question presented.

5.2.4. Technical content

Similarly to Section 5.1, information given above in Section 5.2 did not contain any coding and therefore the technical content will be provided separately in this section. This will be shown for the selection of questions presented earlier.

All coding will be shown for the example presented in Figure 5.2.2a and this is as follows:

- Arrays of available propositions and connectives

```
symbolic_propositions = new Array("","<mi>p</mi><mo>&wedge;</mo><mi>q</mi>","<mi>p</mi><mo>&vee;</mo><mi>q</mi>","<mi>p</mi><mo>&rarr;</mo><mi>q</mi>","<mi>p</mi><mo>&larr;</mo><mi>q</mi>","<mi>p</mi><mo>&harr;</mo><mi>q</mi>");
```

symbolic_propositions is an array containing possible propositions in the symbolic form, listed as the HTML codes. Other arrays contain: connectives in the symbolic form (*symbolic_props* array), connectives in English form (*symbolic_propositions_words* array) and propositions in English form (*propositions* array).

- Number of propositions and connectives

```
pl = propositions.length - 1;
sl = symbolic_propositions.length - 1;
```

pl and *sl* represent the number of propositions and connectives. They are 10 and 5 respectively.

- Random selection of propositions and connectives for the question

```
random_array_indices = displayarray(2,1,pl,0);
p_index = random_array_indices[0];
q_index = random_array_indices[1];
other_symbolic_indices = displayarray(5,1,sl,0);
symbolic_index = other_symbolic_indices[0];
```

random_array_indices is an array containing 2 random numbers between 1 and 10 (10 is the number of propositions). *p_index* is the index 0 value and *q_index* the index 1 of *random_array_indices*. Further, they are used to select two propositions for the question.

other_symbolic_indices is an array containing 5 random numbers between 1 and 5 (5 is the number of available connectives). *symbolic_index* is the first value assigned to that array. It is used to select the connective students are asked to find the translation of as well as to select the symbolic proposition that is a correct answer. The remaining values of an array are used to form distractors presented as the possible options for selection.

- getGrid function to generate English statement for translation and its symbolic representation

```
Grid = getGrid(pl,pl,sl);
for(ip=1;ip<=pl;ip++){
for(isym=1;isym<=sl;isym++){
for(iq=1;iq<=pl;iq++){
Grid[ip][iq][isym] = propositions[ip] + symbolic_propositions_words[isym] + propositions[iq]
+".";
Grid[ip][iq][isym] = Grid[ip][iq][isym].substring(0,1).toUpperCase() +
Grid[ip][iq][isym].substring(1,Grid[ip][iq][isym].length);}}}
```

getGrid(pl,pl,sl) function returns a 3D grid of the size *pl* x *pl* x *sl* with all the entries F (false). The for loop starts at 1st entry in each dimension, increases by 1 each time and continues to run as long as plth or slth entry (depending on the dimension). It assigns propositions and connectives from defined arrays. Then, *toUpperCase()* method converts the first character (of index 0) of the string to uppercase letter and returns it together with the remaining part of the string.

5.3. Applications

The database comprises two questions on applications. Questions within this subtopic are of multiple-response type, but of different styles.

Both questions are about turning the minimum number of playing cards such that it is certain that a specific card is turned over from the available cards. The instructions are the same for each of the questions. However, depending on the question, the selection has to be made when four or six cards are given.

An example of a question on applications is presented in Figure 5.3 below.

Four playing cards are drawn at random from 4 packs. The back sides of the cards are blue, green, red, and maroon, depending on which pack the card was drawn from. The cards are placed on a table with the following facing up:

Jack, green back, ten, blue back.

Turning over as few cards as possible, which card(s) need to be turned over to be certain that, of the four cards selected, every Jack is drawn only from the pack with the blue back side?

☐ blue back

☐ Jack

☐ green back

☐ ten

Figure 5.3: Question on applications, turning over cards from four available cards.

5.3.1. Assumed and tested skills

The questions are dressed in the theme of playing cards. Since only English words are given and translation to the symbolic logic is required, the assumed skills are the same as those described in the Section 5.2.1.

It is also assumed, that students are able to rephrase the statement like in the Figure 5.3, that is 'every King is drawn only from the pack with the red back side' into for example 'if the King is drawn from the pack, then it must have the red back side' without changing its meaning. This statement in symbolic logic is $p \to q$.

Students are tested on the **conditional statements**. Starting with the conditional statement $p \to q$, where *p* is called the hypothesis and *q* is called the conclusion, three variants of conditional statements can be formed. The contrapositive of $p \to q$ is $\sim q \to \sim p$. Using the knowledge on truth values, one can see that both these statements have the same truth tables. Therefore, they are equivalent. The converse of $p \to q$ is $q \to p$ and the inverse of $p \to q$ is $\sim p \to \sim q$. Conditional statements are defined as a tested skill since knowledge of them is required when making a selection of the correct answer.

5.3.2. Objective questions

Multiple-response type questions

The question presented in Figure 5.3 asks to turn over as few cards as possible. Therefore, multiple selection should be allowed and this is possible in the multiple-response type questions. Since students are not informed how many cards they should choose, selecting any amount of cards out of those available is possible. To correctly answer the questions, two cards should be selected when four are drawn at random and three when six is presented.

Different styles

Although the two questions are dressed in the same contest they differ by the provided information. In one question four playing cards placed on the table, whereas in the second, six. This results in the number of cards to be selected by the participants.

Random parameters

The amount of cards placed on the table is fixed for each of the question, being either four or six. However, each time the question is run those cards are chosen at random. This selection is made between four available colours of the back of the card, as well as 13 possible faces, e.g. three, King, Ace, etc. After this selection has been made further selection between them takes place and two or three cards are chosen to complete the statement that is the subject of the question. Where in the middle of the text as well as at the end of the question listed are the drawn cards the order they are presented is randomised.

5.3.3. Feedback

On the feedback screen the original question is followed by the explanation. It will be described on the example presented in Figure 5.3.3. This starts with the analysis of the final statement ‘every King is drawn only from the pack with the green back side’. It is done by setting p for 'this card is a King' and q for 'this card has a green back'. From there follows the claim $p \rightarrow q$, which can be easier deducted if the final statement was rephrased as in Section 5.3.1. In order to make a correct selection of cards students are advised to check the claim and its contrapositive when p is true. Then, each of the selected answers is analysed. It is stated whether the answer is correct or wrong and explained by identifying which conditional statement is being made due to the selection.

Six playing cards are drawn at random from 4 packs. The back sides of the cards are blue, green, red, and maroon, depending on which pack the card was drawn from. The cards are placed on a table with the following facing up:

King, red back, blue back, ten, five, green back.

Turning over as few cards as possible, which card(s) need to be turned over to be certain that, of the six cards selected, every King is drawn only from the pack with the green back side?

Let p be the statement 'this card is a King' and q be the statement 'this card has a green back'. Then the claim (or proposition) being made is $p \rightarrow q$. To decide if this is true or not, you need to check the claim (or proposition) $p \rightarrow q$ and its contrapositive $\sim q \rightarrow \sim p$ when p is true.

C1. Your answer, King, was correct. This is the claim (or proposition) $p \rightarrow q$ being made.
C3. Your answer, blue back, was correct. This is the contrapositive $\sim q \rightarrow \sim p$ of the claim (or proposition) $p \rightarrow q$ being made. For the claim to be true, this card must NOT have a King on the other side!
D1. Your answer, ten, was wrong. This is the inverse $\sim p \rightarrow \sim q$ of the claim (or proposition) $p \rightarrow q$ being made. We do not care what's on the other side of this card.
D2. Your answer, green back, was wrong. This is the converse $q \rightarrow p$ of the claim (or proposition) $p \rightarrow q$ being made. We do not care what's on the other side of this card.

More formally, we can set these possibilities out as a truth table:

p	q	p	$\rightarrow$	q
T	**T**	T	T	T
T	**F**	T	F	F
F	**T**	F	T	T
F	**F**	F	T	F

We see that the statement is false when p = T and q = F. So we need to turn over any Kings and any non-green backed cards to make sure we exclude this possibility.

Figure 5.3.3: Feedback of the question on applications, turning over cards from six available cards.

5.3.4. Technical content

Technical content on applications will be presented in this section. All the coding will be shown on the example presented in Figure 5.3 and this is as follows:

- Random selection of cards to be placed on the table

```
pictures = new  Array("King","Queen","Jack");
nonpictures = new
Array("ace","two","three","four","five","six","seven","eight","nine","ten","joker");
colours = new  Array("blue","green","red","maroon");
```

pictures, nonpictures, colours are arrays storing all possible playing cards to be placed on the table.

```
A = Math.ceil(3*Math.random())-1;          // 0,1,2
B = Math.ceil(11*Math.random())-1;         // 0-10 inclusive
C = Math.ceil(4*Math.random())-1;          // 0,1,2,3
D = (C+1)%4;
```

A, B, C, D are numbers. They are generated using two functions: *Math.random()* and M*ath.ceil()* described earlier, as well as arithmetic operator of modulus that in JavaScript is represented by *%.* Those numbers are then used to define the position of an array's element to be used in the question, e.g. *picture_chosen* is the card chosen from a *pictures* array that is positioned at index *A*, as presented in the code below.

```
picture_chosen= pictures[A];
nonpicture_chosen= nonpictures[B];
colour_chosen= colours[C];
noncolour_chosen= colours[D];
```

- Correct answers and distractors

```
c1=picture_chosen;
c2="<font color="+noncolour_chosen+">"+noncolour_chosen+" back</font>";
d1=nonpicture_chosen;
d2="<font color="+colour_chosen+">"+colour_chosen+" back</font>";
```

c1, c2, d1, d2 define four cards of which the first two are the correct answers and the other distractors. *c1* and *d1* display in the colour that is set up by the users as they represent faces of cards, whereas *c2* and *d2* represent cards placed facing back side up and therefore they are displayed in the colour. This is done using HTML <font> attribute that specifies the colour of the text.

5.4. Connectives

Connectives is an alternative name for the logical operators. In this set of questions conjunction and disjunction are involved.

The database comprises two questions on connectives. Questions within this subtopic are of multiple choice type, but of different styles.

An example of a question on connectives is presented in Figure 5.4 below.

Given the following equation, which statement is always true?

$3a + 6b = 0$

- ○ *a* or *b* is zero or *a* and *b* have opposite signs
- ○ *a* or *b* is zero
- ○ *a* or *b* is zero or *a* and *b* have the same signs
- ○ *a* and *b* are zero
- ○ None of these!
- ○ I don't know!

Figure 5.4: Question on connectives (addition).

5.4.1. Assumed and tested skills

It is assumed that students know what the **conjunction** and **disjunction** are. They are defined in Section 5.1.1. Given two propositions *p* and *q*, conjunction is '*p* and *q*' denoted by $p \wedge q$, whereas disjunction is '*p* or *q*' denoted by $p \vee q$. Conjunction has one clear meaning, i.e. both statements must be true for the conjunction to be true. On the other hand, when statement '*p* or *q*' is given, it can be understood in two ways. The **exclusive** sense is when *p* or *q* but not both can happen (are true) and **inclusive** sense *p* or *q* or both can happen (are true). The feedback (see Figure 5.4.1) discusses this.

~~~~~~Your result~~~~~~

1). Your answer *a* or *b* is zero should have been *a* or *b* is zero or *a* and *b* have opposite signs.

Certainly if *a* = 0, then *b* = 0 as well if the equation is to hold. Similarly if *b* = 0, then *a* = 0 as well if the equation is to hold.

However, the equation can still hold if neither are zero and the terms cancel each other (one positive, one negative). Explicitly we have *a* = -6 *b* / 3 or *b* = -3 *a* / 6.
Either way, *a* and *b* have opposite signs.

Note in ordinary English, the word **or** sometimes means that *a* and *b* cannot be zero at the same time, i.e. it is the *exclusive or*.

However, here *a* could equal 0, so then *b* would be zero too (and vice versa).
The mathematical definition of *or* includes this case, i.e. it is the *inclusive or*.

Figure 5.4.1: Feedback of the question on connectives in Figure 5.4 (addition).

It is also assumed that students are aware of the notation of **equations** with two unknowns (variables). They are expected to know that the values of these variables should be found in order for the equation to be true.

Combining the assumed skills together, students are tested on deciding what values the unknowns should take for the equation to be true.

5.4.2. Objective questions

Multiple choice type questions

Students have to choose one of the possible 6 options. Next to the options ‘None of these!’ and ‘I don’t know!’ that are present in every MC type question there may display a correct answer or a selection of available distractors. These involve connectives *‘and’* and *‘or’*, as well as the information about the sign, i.e. the same or opposite, describing the variables.

Different styles

The equations are present in both questions. They are composed of two terms (products of a coefficient and an unknown) connected by either addition or multiplication sign, depending on the question.
~~~~~~

Random parameters

The coefficients of unknowns are defined by the numbers. They are chosen at random every time the question is run. Four out of six options for the answer are also selected individually every time.

5.4.3. Feedback

The feedback screen (Figure 5.4.3) includes the question, submitted and correct answers. Then, the specific feedback follows. It starts with the information of two possible interpretations of the word '*or*'. Both, the ordinary English and mathematical definitions of the connective are described to support the correct answer.

Given the following equation, which statement is always true?

$6a \times 5b = 0$

~~~~~~Your result~~~~~~

3). Your answer *a* and *b* are zero should have been *a* or *b* is zero.

Certainly if *a* or *b* is zero then the equation is true ... but what about if both of them are zero? This depends on the precise mathematical meaning of the word *or*.

In ordinary English, the word **or** sometimes means that *a* and *b* cannot be zero at the same time, i.e. it is the *exclusive or*.
However, here *a* could equal *b* and they could both be zero.
The mathematical definition of *or* includes this case, i.e. it is the *inclusive or*.

Related material

0 out of 1
Your answer is incorrect.
The only way to get 0 is if one (or both) of the numbers *a*, *b* is (are) zero.

Figure 5.4.3: Feedback of the question on connectives (multiplication).

5.4.4. Technical content

The questions on connectives do not involve any complicated coding. It includes the selection of two numbers using the *displayarray()* function from the range between 2 and 10 that is described in Section 4.1.4. The correct answer as well as distractors are pre assigned and are the same every time the question is run.

6. Linear programming

Linear programming is an area of mathematics and a process of decision making leading to finding the optimal solution. To formulate a linear programming problem it is necessary to define the variables, determine the constraints and the objective function. Then it can be either solved using the graphical method (that was a subject of constructing questions presented further in the chapter) or the simplex method (not discussed here).

The database includes questions on feasible region and optimisation. Therefore, subchapters presented will be organised according to these two topics.

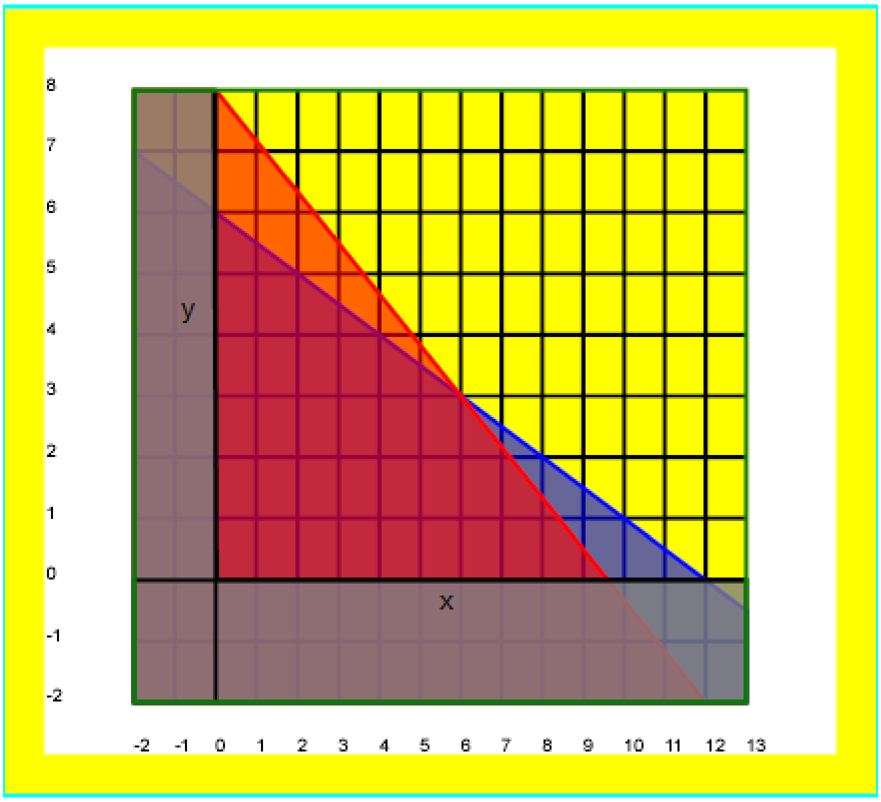
6.1. Feasible region

Sketching the feasible region is a graphical method used to find the optimal solution. The feasible region is the region containing the collection of points satisfying the constraints defined by the inequalities in the two-variable problems.

The database comprises of 11 questions on feasible region. Questions are of the multiple choice type, but of a different style. They differ from each other by the matching that has to be performed. It involves two of the following: graph, inequalities and scenario.

An example of a feasible region question is presented in Figure 6.1 below.

Connie is given a graph showing some of a feasible region (shown yellow) in a linear programming problem. Points on either line are feasible but non-feasible regions have been shaded out. Which of the sets of (possibly unsimplified) inequalities below represents the yellow region on the graph shown?



○ $3x + 6y \leq 36, \quad 5x + 6y \geq 48, \quad x \geq 0, \quad y \geq 0$

○ $3x + 6y \leq 36, \quad 5x + 6y \leq 48, \quad x \geq 0, \quad y \geq 0$

○ $3x + 6y \geq 36, \quad 5x + 6y \leq 48, \quad x \geq 0, \quad y \geq 0$

○ $3x + 6y \geq 36, \quad 5x + 6y \geq 48, \quad x \geq 0, \quad y \geq 0$

○ None of these!

○ I don't know!

Figure 6.1: Question on feasible region, given graph find matching inequalities.

6.1.1. Assumed and tested skills

A number of related to feasible region concepts can be distinguished. They form the set of assumed skills and differ between the questions. As presented in the Figure 6.1, it is expected that students know that a **linear equation** has a form $ax + by = c$ where *a, b* and *c* are positive numbers and its graph is a **straight line**. Then, it is known that the line represents the set of solutions of the equation. Furthermore, the straight line divides the 2D plane into two half-planes.

Students are tested whether they know which half-plane represents $ax+by<c$ and which $ax+by>c$. They are given a collection of inequalities and the graphical representation of them. Therefore, students are tested on **matching** either the inequalities with the indicated region on the graph or vice-versa. Two questions present students with scenarios. In these cases they are tested on identifying the inequalities when the information is given by text.

6.1.2. Objective questions

Multiple choice type questions

Students are asked to select one of six available answers. They include answers such as ‘None of these!’, ‘I don’t know!’, distractors and correct answer. Distractors depend on the question and can either be different region’s colours from the graph, or inequalities with wrong inequality signs or with wrong coefficients.

Different styles

Eight questions give students graphical representation of the constraints and ask for the corresponding inequalities to match indicated region. They are clones of each other and differ by the specified area of the graph or the coefficients used in the distractors. Another question is reverse-engineered and asks for the region matching the given inequalities. The remaining two questions provide students with the scenarios for which inequalities must be matched.

Random parameters

The coefficients of the inequalities are randomly selected. They are also used to automatically generate the graph for the purpose of the questions. All the numbers in the scenarios are chosen at the time when questions are run and used to generate the inequalities. In the questions with inequalities as the answers, the selection of correct answer and distractors is chosen from the available ones and allocated to the first four answer options in a random order.

6.1.3. Feedback

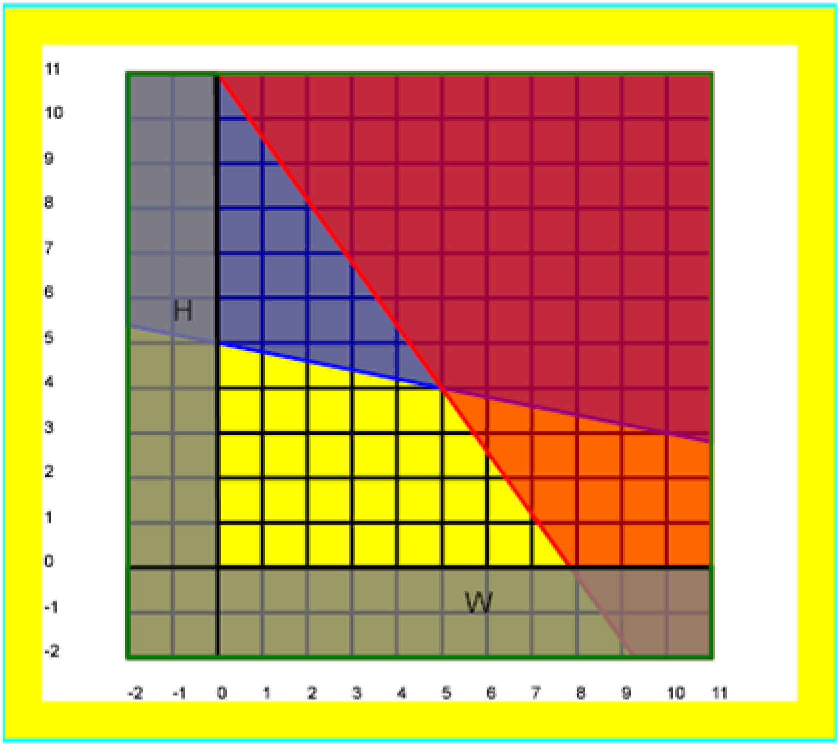
The feedback screen starts with the repeated question. After the wrong answer is submitted, the correct answer and its explanation follow. As presented in Figure 6.1.3, both inequalities of the $ax + by \leq c$ form are rearranged into $y \leq -{}^{a}\!/_{b}\,x + {}^{c}\!/_{b}$ and later into the linear equations of the form $y = -{}^{a}\!/_{b}\,x + {}^{c}\!/_{b}$ being the straight line and the boundary of the region.

A city trader is considering selling unit trusts comprising *H* shares in Hippo International and *W* shares in Wild Boar Associates. Costs are as follows:

- Hippo International shares cost ¥5 each and Wild Boar Associates shares cost ¥1 each. Research has shown that the maximum amount the investor is willing to invest per unit trust is ¥25.
- There are also financial agent charges per unit trust as follows: Hippo International shares cost 20 cents each to buy and Wild Boar Associates shares cost 28 cents each to buy. Total financial agent charges are limited to 220 cents.

Which set of inequalities describes the feasible region?

The feasible region is shown in yellow below:



~~~~~~Your result~~~~~~

2). Your answer $28W + 20H \geq 25, \quad W + 5H \geq 220, \quad W \geq 0, \quad H \geq 0$

is not a region shown on the graph.
Your answer should have been

$W + 5H \leq 25, \quad 28W + 20H \leq 220, \quad W \geq 0, \quad H \geq 0$

The first inequality can be rearranged to give: $H \leq -\frac{1}{5}W + 5$

Hence the equation for the boundary blue line can be written as: $H = -\frac{1}{5}W + 5$

Similarly the second inequality can be rearranged to give: $H \leq -\frac{7}{5}W + 11$

Hence the equation for the boundary red line can be written as: $H = -\frac{7}{5}W + 11$

The feasible region is the area in the first quadrant and **below or on** the blue line and **below or on** the red line.

Figure 6.1.3: Feedback of the questions on feasible region, given scenario find matching inequalities.

6.1.4. Technical content

All the numbers displayed in the questions are created using operations on numbers generated by the *displayarray()* function described in Section 4.1.4. They are then used to produce the inequalities. The coding for the correct answer to the question presented in Figure 6.1.3 is as follows:

```
c1="<PARAM NAME='eq' VALUE='<math>%START_MSTYLE%"+displayquadform(y,x,coeffs1,2)+"<mo>&le;</mo><mn>"+x0*y1+"</mn><mtext>,</mtext><mtext> </mtext><mtext> </mtext>"+displayquadform(y,x,coeffs2,2)+"<mo>&le;</mo><mn>"+factor*x0*y2+"</mn><mtext>,</mtext><mtext> </mtext><mtext> </mtext><mi>"+x+"</mi><mo>&ge;</mo><mn>0</mn><mtext>,</mtext><mtext> </mtext><mtext> </mtext><mi>"+y+"</mi><mo>&ge;</mo><mn>0</mn>%END_MSTYLE%</math>' ></APPLET>"
```

Function *displayquadform*, stored in *brunel_algebra,* returns the MathML syntax for a quadratic form. Its parameters *(y,x,coeffs1,2)* define the function for display. It has two unknowns *y* and *x*, defined for each scenario, e.g. *H* for Hippo and *Z* for Zebra. Coefficients of the first inequality are stored in *coeffs1* array. The last parameter defines the number of terms or the quadratic form on top of the constant term; it is *2* since only the first power terms are needed. Completing the first inequality of the correct answer is the less or equal to HTML entity code *≤* and the limit calculated by the product of two numbers *x0* and *y1*. To separate each inequality another HTML code is used, i.e. * *. The remaining part of the code includes three other inequalities of the correct answer.

6.2. Optimisation

Finding the optimal solution is a step further after defining the feasible region. It is also the final step in solving the linear programming problems. On top of the constraints, the objective function is needed to find the optimal value specified in the question.

The database comprises of three questions on optimisation. Questions are of the responsive numerical input and 2 numerical input types, but of a different style. They differ from each other by what should be found and also by the possibility of the solution being degenerate in one question and unique in two remaining questions.

An example of an optimisation question is presented in Figure 6.2 below.

A health authority is testing a disease treatment that requires a combination of drugs and hospitalisation. Patients are divided into groups A and B, P_A and P_B being the numbers of patients in groups A and B respectively. Costs are as follows:

- Drug A costs $5,000 per course and drug B costs $3,000 per course. Total drug cost is limited to $35,000.
- Drug A takes 25 days in hospital and drug B takes 25 days in hospital. The total number of days in hospital is limited to 225.

Average recovery rates for each treatment are: 42.00 % for group A and 8.70 % for group B. What is the maximum expected number of patients that will recover if the health authority uses its resources properly?

Input your answer correct to 2 decimal places.

Figure 6.2: Question on optimisation, RNI.

6.2.1. Assumed and tested skills

It is assumed that students can assimilate all the information that is presented to them by the question in order to define variables, constraints and the objective function. As with the feasible region questions, it is also assumed that they know concepts of linear equations, that they can sketch their graphs as well as those of inequalities.

Students are tested on finding the values of *x* and *y* satisfying all the constraints and resulting in the optimal solution which value they are also tested on finding. “The optimal solution of a linear programming problem, if it exists, will occur at one or more of the extreme points (vertices) of the feasible region.” (Hebborn, 2000, p. 155) It can be one or two corners of the feasible region or a line joining them. A way to determine them is to use the ruler or vertex method. Questions do not state the method to be used, therefore tested skills cannot be defined in any more details.

6.2.2. Objective questions

Responsive numerical input type questions

These questions ask for the optimal value to the problem. All of the six different scenarios result in the graph containing four vertices: at the origin, at the x-axis, at the y-axis, and in the first quadrant. Therefore, these corners are possible answers and between them it is decided which is the correct answer and which points could be used as distractors. Since the coordinates (0, 0) of the origin always produce value of zero, it is never the correct answer and therefore not used as a possible distractor.

2 numerical input type question

Students are asked to input values of two variables that would result in the optimal solution.

Different styles

Questions differ by the given guidelines, two ask for the optimal solution, whereas one asks for the values of two variables that would produce the optimal solution. Moreover, in two questions the optimal solution is given by a unique vertex, whereas in the third it is possible that the solution would be degenerate as more than one point optimises the problem.

Random parameters

All the numbers used in the questions are randomly selected. However, the numbers used in the questions are chosen in the way so that the intersection points always have integer coordinates. Otherwise, nonsense like 6.4 caribous and 2.8 zebras could arise. Scenarios presented to students are randomly chosen from seven available. Objects used in some questions, e.g. electronic products, animals, companies' names, are selected from the available list as well as currency type used in some questions.

6.2.3. Feedback

The feedback screen (Figure 6.2.3) consists of the constraints, objective function, and the graph with a feasible region in yellow as well as white line representing the objective function. Then the correct answer is given. The page finishes with the comment on the given answer, i.e. both answer were wrong, both answer were correct, or that only one answer was correct.

An electronics company produces printers (p) and DVD players (D). Demand exceeds production for both products. Individual item times are as follows:

- printers take 6 minutes to package and DVD players take 2 minutes to package. Total packaging minutes are limited to 36 per packer.
- printers take 30 hours to produce and DVD players take 30 hours to produce. Total production hours are limited to 300 per robot.

The profit for each product is: £100.00 per printer and £49.00 per DVD player.
What values of *D* and *p* should the company choose to maximise the profit that one packer and one robot can generate?

The feasible region (shown yellow in the graph) is given by

$2D + 6p \leq 36$

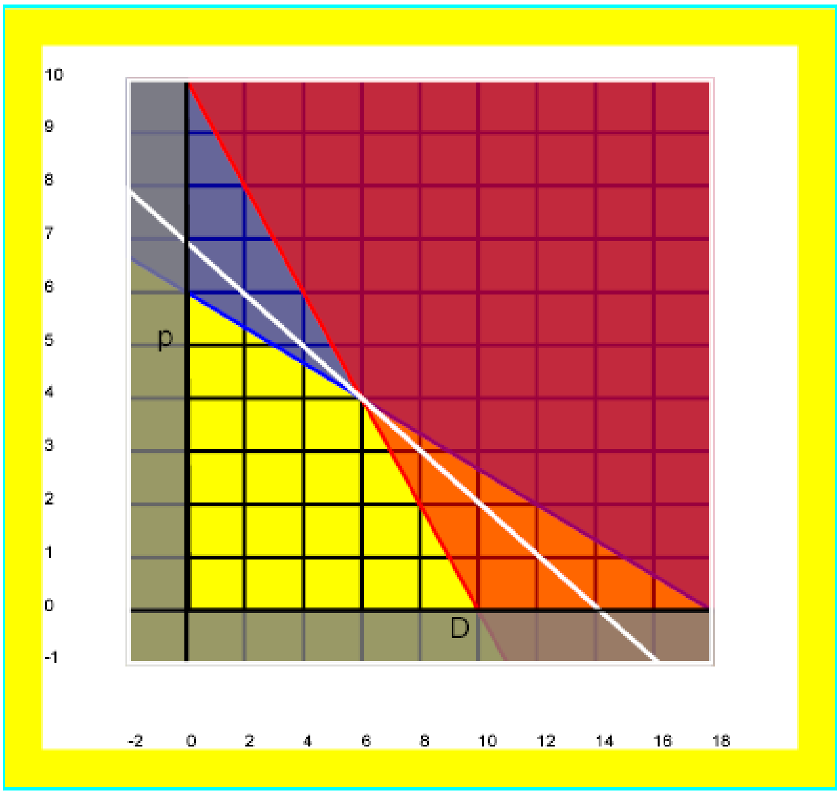
$30D + 30p \leq 300$

$D \geq 0$

$p \geq 0$

and the objective function *J* (shown maximised by a white line in the graph) is given by

$J = 49D + 100p$



The optimum point is where the white line touches the yellow region. This happens when p = 4 and D = 6. Although not asked for, the value of *J* at this point is 694.00

~~~~~~Your result~~~~~~
Your answers 2 and 3 were wrong. The correct answers were 6 and 4.

Figure 6.2.3: Feedback of the questions on optimisation, 2NI.

6.2.4. Technical content

The coding below is described and partially shown on the example presented in Figure 6.2.3.

The values of variables (*x0, y0, y1, y2, factor, scenario_objective_factor, alpha, beta*) are generated by the *displayarray()* function described in Section 4.1.4. They are used in the randomly selected scenario, constraints, objective function, as well as in the SVG diagram producing the graph displayed on the feedback screen.

- Choosing the scenario

```
scenario = Math.ceil(7*Math.random())-1;
```

scenario is a number generated using *Math.ceil()* and *Math.random()* functions (described in Section 3.1.4) and takes value between 0 and 6 inclusive. Depending on what value is selected, the corresponding scenario displays.

- Description of the scenario

```
if(scenario == 4){
itemsoldarray = new Array("wild boar","carribou","hippo","giraffe","zebra","elephant","yak",
"buffalo","moose");
itemsoldindex = Math.ceil(itemsoldarray.length*Math.random())-1;
itemsold1 = itemsoldarray[itemsoldindex%itemsoldarray.length];
more = Math.ceil(Math.random()*(itemsoldarray.length-1));
itemsoldindex +=more;
itemsold2 = itemsoldarray[itemsoldindex%itemsoldarray.length];
y = itemsold1.substring(0,1);
x = itemsold2.substring(0,1);

QuestionText%QUESTION.NUMBER% = "A zoo keeps "+y+" "+itemsold1+"s  and "+x+" "+itemsold2+"s. Individual animal overheads are as follows:<ul><li>"+itemsold1+"s use "+x0+" tonnes of bedding per month and "+itemsold2+"s use "+(y1-y0)+" tonnes of bedding per month. Total bedding tonnage is limited to "+x0*y1+" per month.</li><li>"+itemsold1+"s eat "+factor*x0+" tonnes of food per year and "+itemsold2+"s eat "+factor*(y2-y0)+" tonnes of food per year. Total food tonnage is limited to "+factor*x0*y2+" per year.</li></ul><p>"
```

*QuestionText%QUESTION.NUMBER% += "Average visitor times (person minutes per day) for each animal type are: "+(scenario_objective_factor*alpha).toFixed(2)+" minutes per "+itemsold1+" and "+(scenario_objective_factor*beta).toFixed(2)+" minutes per "+itemsold2+".
What values of <i>"+x+"</i> and <i>"+y+"</i> should the zoo choose to maximise the visitor person minutes per day for both animal enclosures?<p>" }*

When *scenario* equals to 4, the above code is executed and scenario on zoo animals is displayed to students. *itemsoldarray* is an array holding nine names of animals from which two are randomly chosen and used in the scenario. *itemsoldindex* and *more* are numbers that are used to define the indexes of the animals. *itemsold1* and *itemsold2* are the names of animals selected at random from an array holding the list of them. *y* and *x* are obtained using the *substring()* which extracts part of the string. In the coding above, the extraction starts at the index 0 and stops at 1 (not including the character at the index 1). As the result the first characters of animals' names are returned.

After all the random parameters are defined they are placed into the text of the scenario. Thes include both, parameters defined for all the scenarios, i.e. *x0, y0, y1, y2, factor, scenario_objective_factor, alpha, beta,* as well as those specific for the scenario. i.e. *x, y, itemsold1, itemsold2.*

7. Graph theory

Graph theory is the study of graphs which consist of a non-empty but finite set of vertices connected by edges. The number of concepts associated with graph theory can be distinguished. Different graph types include: undirected graph, directed graph, simple graph, multigraph, weighted graph, tree, etc. They can be represented graphically, by matrices, or by vertex/edge sets. These and other definitions were the subject for constructing questions and will be described in the ‘assumed and tested skills’ paragraphs.

7.1. Adjacency matrix

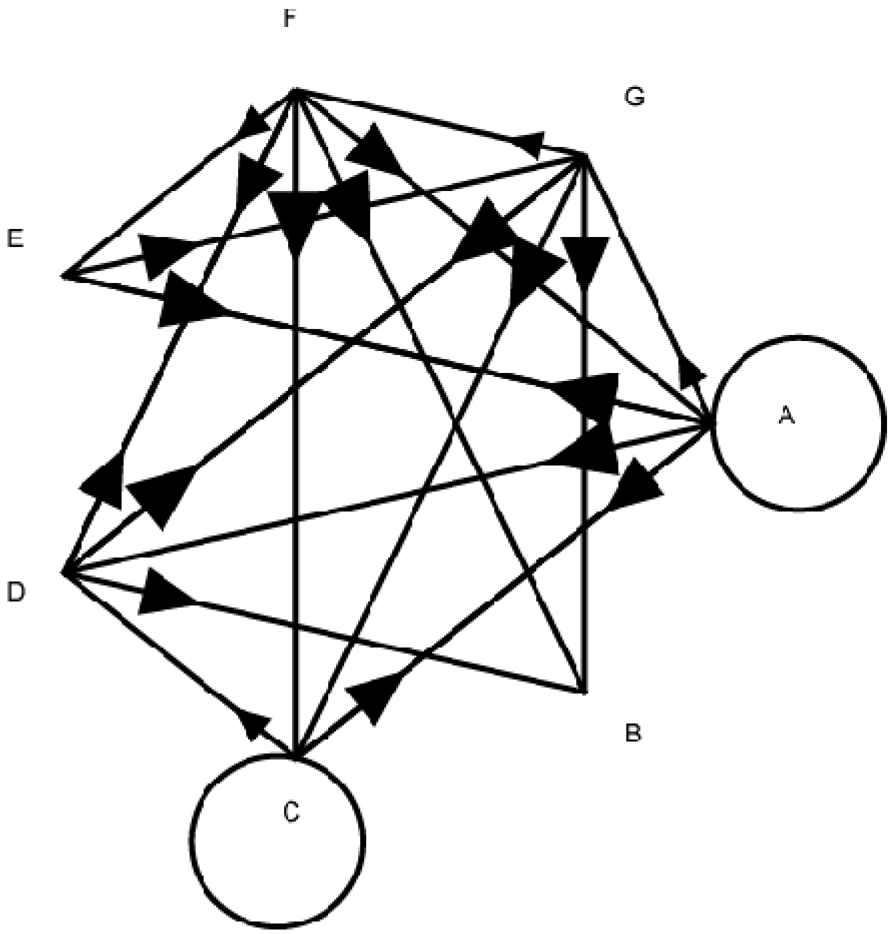
The adjacency matrix is one of the possible representations of the graphs. “Each row and each column of the matrix represents a vertex of the graph” (Hebborn, 2000, p. 37) and are usually labelled with the letters. “The numbers in the matrix give the number of edges joining the pair of vertices.” (Hebborn, 2000, p. 37) An edge connecting one vertex to itself is called a loop, is regarded as two edges and therefore results in 2’s in the diagonal of the matrix by the corresponding vertex.

The database comprises of two questions of different type, i.e. multiple choice and responsive word input, and of a different style. The question either asks to select the adjacency matrix matching the given graph or to indicate an edge of the digraph that does not match with the given matrix.

An example of a question on an adjacency matrix is presented in Figure 7.1 below.

Consider the following adjacency matrix and the digraph:

	A	B	C	D	E	F	G
A	2	0	1	1	1	0	1
B	0	0	0	0	0	0	0
C	1	0	2	1	0	0	0
D	0	1	0	0	0	1	1
E	1	0	0	0	0	0	1
F	1	1	1	0	1	0	0
G	0	1	1	1	0	1	0



The given adjacency matrix has a mistake in it and does not match the digraph shown.

Please input the edge of the digraph that is missing or added in the adjacency matrix . *For example, if you think the problem occurs at the highlighted position* $\boldsymbol{a_{3,4}}$ *then enter your answer as* ***CD***.

Figure 7.1: Question on an adjacency matrix, RWI.

7.1.1. Assumed and tested skills

It is assumed that students know what a graph is. As mentioned at the beginning of this chapter it is a collection of vertices (points) and edges (lines connecting those points). Although not tested on the properties of different types of graphs it is assumed that students know that the multigraph is a graph with multiple edges connecting different vertices or the same vertex and then it is called a loop. Digraphs, on the other hand, are graphs with directed edges. Edges are described using the names of the vertices they connect. As in the case of an undirected graph the order of the labels does not matter, it is important when specifying the edges of the digraph. It is also assumed that students know what the adjacency matrix is as well as its upper/lower triangular matrix elements in order to be able to understand the feedback. Graphs in both questions have loops associated with them, therefore it is important that students know that one loop is considering as two edges and 2 should be used in the adjacency matrices to represent each of them.

Questions are indirectly testing students' ability on finding the adjacency matrix representation of the graph when it is given graphically or vice versa.

7.1.2. Objective questions

Multiple choice type questions

Students are presented with a multigraph and they are asked to indicate the matching adjacency matrix. The possible answers for selection consist of the correct answer, distractors, 'None of these!', 'I don't know!'. Available are four distractors for display: the entries of the matrix where corresponding vertices are connected are marked with 1's, even if there is more than one edge connecting them; as explained earlier, loops represent two edges, however, 1's are used in the diagonal instead of 2's; the graphs used are symmetric, however, displayed matrix is anti-symmetric with random entries; adjustments have been made to all values in the matrix.

Responsive word input type question

The question (Figure 7.1) gives students the adjacency matrix and the matching digraph with one exception. One entry of the matrix does not match the edge in the graph. Participants have to input, in the provided box, the edge of the digraph that is missing or added in the adjacency matrix. The only time when students are told what exactly they did wrong is when they type the correct answer backwards and as a result of giving the wrong answer.

Different styles

As mentioned already each of the two questions asks for different things to be found. Since the questions are not clones of each other they cannot be compared showing the differences.

Random parameters

The adjacency matrices are randomly generated. However, before it can be produced, N, the number of vertices has to be randomly chosen.

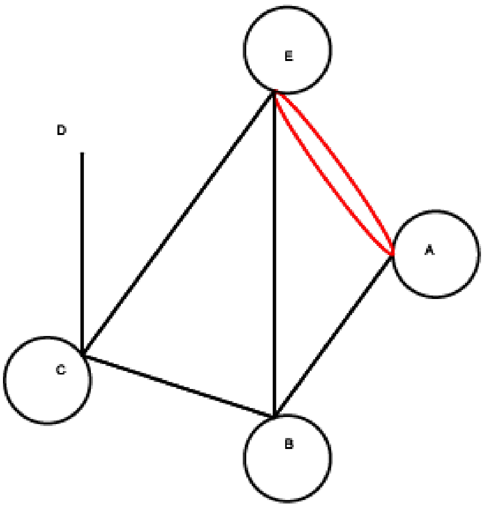
For the question in Figure 7.1 N is decided at random, whether the correct answer will display as one of the answers for selection, or four distractors will be used.

For the question in Figure 7.1.3 the edge that causes a mismatch between adjacency matrix and the digraph is also randomly selected.

7.1.3. Feedback

As presented in the Figure 7.1.3, the feedback screen shows the graph that was a basis of the question and the matching adjacency matrix has to be chosen. Due to the fact that graphs and matrices take a lot of space all the possible answers are not presented, but only the submitted and correct matrices are shown. The feedback suggests on how the question could be attempted to save some time when answering the question next time. Since the distractors are coded and there is an explanation of each of them, this is given to the students as a final comment.

Brendan is given the following graph with double edges shown in red for clarity.



Find the adjacency matrix representing the above graph.

~~~~~~Your result~~~~~~

1). Your answer

	A	B	C	D	E
A	1	1	0	0	1
B	1	1	1	0	1
C	0	1	1	1	1
D	0	0	1	0	0
E	1	1	1	0	1

should have been

	A	B	C	D	E
A	2	1	0	0	2
B	1	2	1	0	1
C	0	1	2	1	1
D	0	0	1	0	0
E	2	1	1	0	2

To find the corresponding adjacency matrix, you should look carefully at every edge to see if it exists in the given graph (or multi-graph). If the graph is undirected, and hence is symmetric (i.e. for $1 \le i \le n$ and for $1 \le j \le n$, $\mathbf{A}_{i,j} = \mathbf{A}_{j,i}$) you should only look at the **upper triangular** or the **lower triangular** part of the matrices (so that it can save you time when answering this question).

By implementing these procedures properly, the answer you should obtain is:

	A	B	C	D	E
A	2	1	0	0	2
B	1	2	1	0	1
C	0	1	2	1	1
D	0	0	1	0	0
E	2	1	1	0	2

Related material

0 out of 1

Your answer is incorrect.

You have the right idea, but you failed to realise that loops are represented by 2's in the diagonal as well as that more than one edge can connect the same pair of vertices, which would enable 2's to appear in the incidence matrix.

Figure 7.1.3: Feedback of the question on adjacency matrix, MC.

7.1.4. Technical content

The coding below is shown on the example presented in Figure 7.1.3 and this is as follows:

- Selection of the number of vertices

```
n = displayarray(1,4,9,0);
```

- Generation of the *n* x *n* matrix for the correct answer

```
A = getrandommatrix(n, n, 0, 2, 1);
// A has to have 0 or 2's down the diagonal
for (i = 1; i <= n; i++){
if(A[i][i] >= 1){A[i][i] = 2;
  }
}
```

getrandommatrix() is a function in *brunel_linearalg* which in this case creates matrix *A* with *n* rows and *n* columns. Allowed elements are integers between 0 and 2. The last argument (1) of the function allows 0's to be present, and so not every vertex has to be connected with every other.

The entries along the diagonal represent connectivity of vertices to themselves. Therefore, there are just two possibilities: 2's if there is a loop, or 0's if there is no loop. When the matrix was generated 1's were allowed for allocation. Those on the diagonal now have to take the value 0 or 2. It has been decided to rewrite those with values greater or equal to 1 with 2 using one line of code: *if(A[i][i] >= 1){A[i][i] = 2;}.*

- Generation of the matrix for distractor 1 selected in Figure 7.1.3, has 1's in the diagonal in the place of 2's in the matrix of the correct answer

```
D1 = getrandommatrix(n, n, 0, 2, 1);
// D1 has to have 0 or 1's down the diagonal
for (i = 1; i <= n; i++){
  for (j = 1; j <= n; j++){
    D1[i][j] = A[i][j];
    if(D1[i][j] ==2){D1[i][j]=1};
  }
}
```

D1 is a matrix generated in the same way as the matrix *A* above. It then has assigned the same values as matrix *A* (*D1[i][j] = A[i][j];*) using the *for* loop and only the values at the diagonal are changed. Those of 2 take values 1 using *if* loop: *if(D1[i][j] ==2){D1[i][j]=1};}*. Other distracters encode suitable mal-rule in a similar way.

Since in the question students are given symmetric graphs, and those randomly generated are not symmetric then they have to be amended by setting *A[i][j] = A[j][i]* in a suitably nested *for* loops that make the upper and lower triangular matrices equal.

7.2. Degree

The degree of a vertex is the number of edges connected to that vertex. Graphs can be described using the degree sequence, i.e. a nonincreasing ordered list of vertex degrees. When dealing with the digraph, two types of the degree can be distinguished: indegree and outdegree. As the names suggest the first one represents the number of edges directed towards the vertex, and the second directed outwards.

The database comprises of eight word input type questions on degree sequence (Figure 7.2) and five numerical input type questions on degree/indegree/outdegree (Figure 7.2.3). Questions in each of these two groups are of different style.

An example of a question on degrees is presented in Figure 7.2 below.

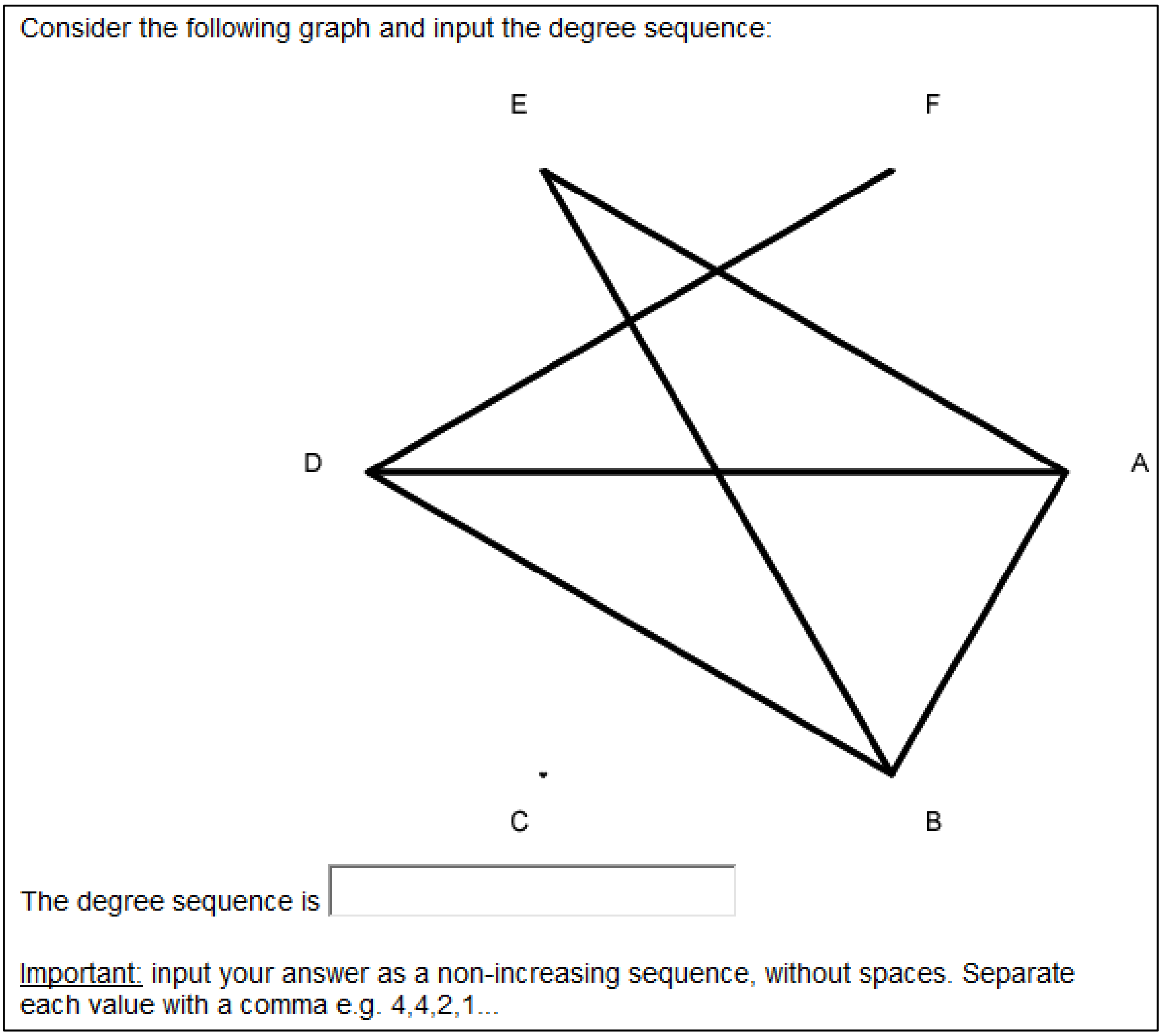


Figure 7.2: Question on degree sequence, WI.

7.2.1. Assumed and tested skills

As described in the Section 7.1.1, it is assumed that students know the definitions/concepts of: graph, vertex, edge, loop, multigraph, digraph, adjacency matrix. It is also assumed, that they know what the network matrix and the isolated vertex are. A network matrix is a name for the matrix representation of a weighted graph, also called network. Weights (numbers) associated with edges characterise this type of a graph. An isolated vertex is a vertex of degree zero, to which there is no edge attached. The final but very trivial assumed skill is that participants can perform very simple addition, e.g. 2 + 2 + 1 + 1 + 0 = 6.

Students are tested on their knowledge on degrees, degree sequence, indegrees, and outdegrees.

7.2.2. Objective questions

Word input type question

Students are presented with graphs, given in the matrix representation or graphically. They have to input the degree sequence in the input box provided. However, it has to be of a specific format and instructions about it are given at the end.

Numerical input type question

Questions present matrices of different type and ask for a degree, indegree, or outdegree of a vertex. A number is expected to be typed in in the input box.

Different styles

The questions on finding the degree sequence of a given graph are clones of each other. A half of the questions provides students with graphical representation of the graph, whereas, others give adjacency matrix. The questions are differentiated further by the type of the graph. It is simple and connected, simple with isolated vertex, multigraph with loops, or multigraph without loops.

The questions on finding the degree/indegree/outdegree of the vertex are also cloned. In all the cases they present matrices, but of different type. These include network matrix of an symmetric graph, adjacency matrix, network matrix of a digraph.

Random parameters

The size of the matrix (number of vertices) is randomly chosen. When the matrices are generated, their entries are also randomly selected. This includes a different number of edges connected to each vertex and weights when talking about the network.

7.2.3. Feedback

The explanation of the solution (see Figure 7.2.3) starts with some theoretical information relevant to the question. It is followed by the instruction on what should be done in order to obtain the correct answer.

Consider the following network matrix and input the indegree of B:

	A	B	C	D	E
A	-	2	3	-	2
B	1	-	2	3	-
C	2	-	-	1	3
D	4	3	3	-	3
E	-	1	-	3	-

~~~~~~Your result~~~~~~
Your answer, 6, should have been 3.

You have been asked to find the **indegree** of B. You should note that you were given a network matrix of a *digraph*, where the letters on the left represent vertices which connected edges are directed outward and letters on the top represent vertices towards which connected edges are directed.

Recall also that any non-zero entry of the network matrix represents one edge in the graph.

Counting the number of non-zero entries in the column labelled by B you should get the correct answer **3**.

Figure 7.2.3: Feedback on indegree of the vertex, WI.

7.2.4. Technical content

The coding below is shown on the example presented in Figure 7.2.3 and this is as follows:

- Assigning weights to the network matrix given the adjacency matrix

```
for (i = 1; i <= n; i++){
for (j = 1; j <= n; j++){
 if(A[i][j] == 1){
    weights[i][j] = A[i][j]*Math.ceil(Math.random()*4);
}}}
```

Matrix *A*, generated in the earlier stage of the coding, in a similar way to the one described in Section 7.1.4, is an asymmetric matrix with elements being 0's or 1's showing the connectivity of vertices. This matrix is then used to assign weights. To do that, *for* loops on *i*'s and on *j*'s are used. In the executed code *if* statement is present. Entries of the matrix *A* equal to 1 are multiplied by the number generated using the *Math.ceil* and *Math.random* (described in Section 3.1.4). In this way the network matrix *weights* is generated.

- Generation of the list of indegrees

```
indegreelist = new Array(n);
for(i = 1; i <= n; i++){
  indegreelist[i-1] = 0;
  for(j = 1; j <= n; j++){
    indegreelist[i-1] += (W[j][i]-0);
  }
}
```

indegreelist is an array holding the indegrees of each vertex. They are assigned using the two *for* loops on *i*'s and on *j*'s. Each time the first loop is executed it calculates the indegree for each vertex. For *i=1* indegree of the first vertex is made equal to 0. Then, each time *j* increases by one, the value of the matrix *W[j][i]* is added (matrix *W* was equated to the matrix *A* earlier in the coding). The first index of the matrix tells from which vertex, i.e. *j*, the edge is connected and the second-into which, i.e. *i*. Therefore, since *i* stays constant until *j* reaches the last value, i.e. *n*, the indegree of vertex *i* is calculated.

7.3. Edge and vertex sets

As the names suggest, the vertex set is the set of all vertices of a graph, whereas the edge set is the set of all the edges of a graph.

The database comprises of five word input type questions with a pop out message asking to double check the given answer. There are two questions on the vertex set (Figure 7.3) and three on the edge set (Figure 7.3.3). Questions in each of these two groups are of different style.

An example of a question on the vertex set is presented in Figure 7.3 below.

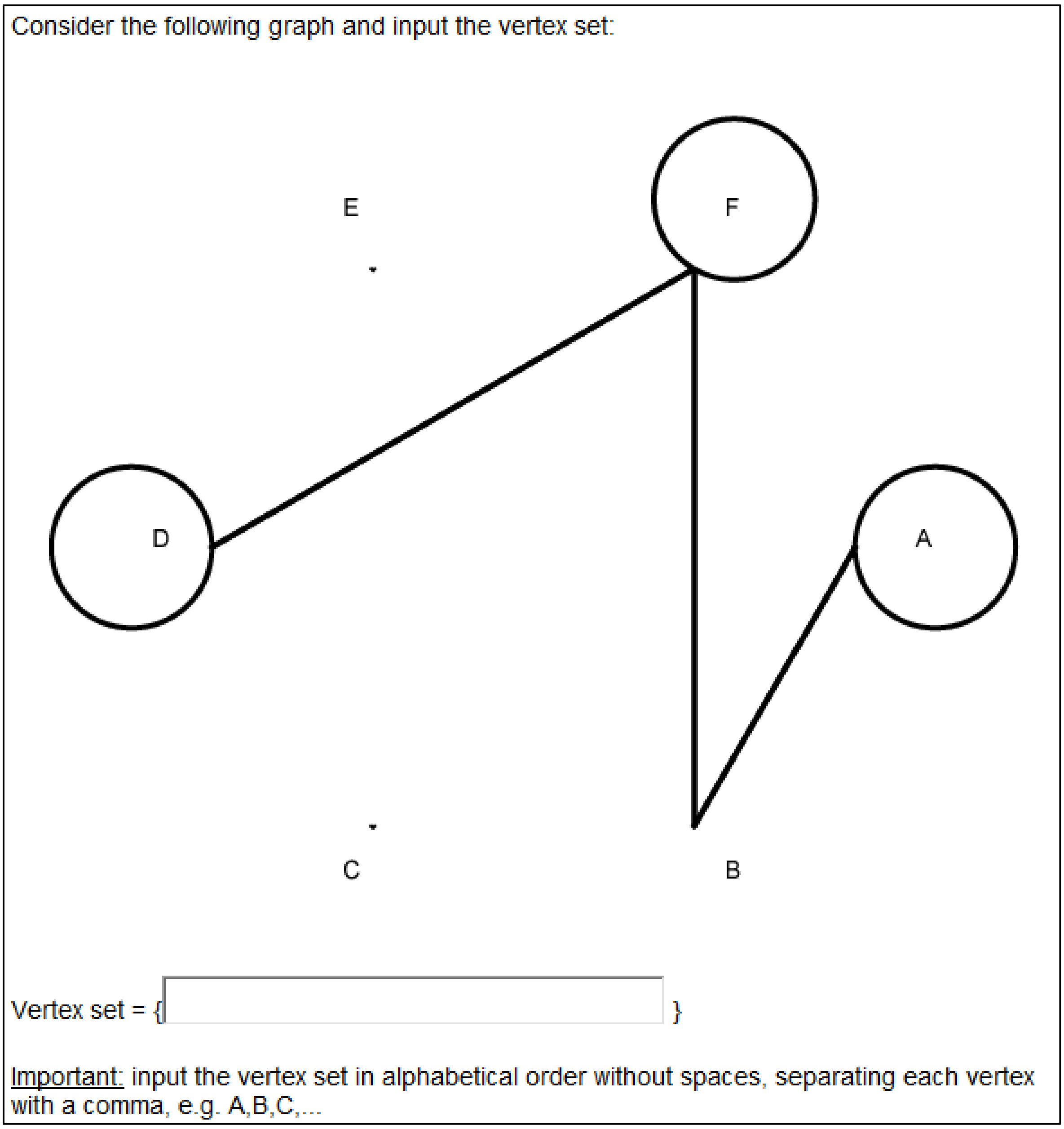


Figure 7.3: Question on the vertex set, WI+check.

7.3.1. Assumed and tested skills

It is assumed that students know the definitions of graph, vertex, edge, loop, digraph (described in the Section 7.1.1) and isolated vertex (described in the Section 7.2.1).

Students are tested on their ability on reading the graph and listing the vertices and the edges. Different types of graphs in each of the questions allow determining those skills depending on the graph.

7.3.2. Objective questions

Word input + check type question

Students are presented with graphs given graphically. They have to input the list of the vertices or edges in the input box. The curly brackets are provided to exclude the notational mistakes and allowing concentration on the tested skills. The answer has to be of a specific format which description is given at the end of the question.

Regardless of the correctness of the submitted answer the message box appears asking to check if the answer is in the required format.

Responsive word input + check type question

When listing the edges of a directed graph a number of common mistakes can be listed. Two of them have been coded and when made by students the directed feedback pointing out this mistake displays. This is the case when every edge is written in the wrong direction, or when all of the edges are listed in alphabetical order when at least one edge does not fit this pattern. All the other aspects of the question are the same as for the word input + check type question described above.

Different styles

The questions are clones of each other. Two of them ask for the vertex set and three for the edge set. Questions within each group differ by the type of a graph. Those on the vertex set are either connected or disconnected. This allows detection of students who would not include isolated vertices in their responses. Questions on the edge set include digraphs, graphs with loops or simple graphs. This, on the other hand, tests students' ability on listing directed edges, loops, or undirected, single edges. By this organisation of the questions it is easier to define missing knowledge when questions are wrongly answered.

Random parameters

Graphs are randomly chosen each time the questions are displayed. Randomisation includes the different number of vertices, different degrees of them, as well as positioning and quantity of the loops and isolated vertices.

7.3.3. Feedback

The explanation of the solution (see Figure 7.3.3) starts with the definition of the edge set and the note on what is different about the digraph compared to the non-directed graph. It is followed by the comment to the instructions given on the format of the answers and how it should be applied for this specific question in order to obtain the correct answer.

Input the edge set in alphabetical order, each vertex being separated by a comma only, e.g. AA,AB,AC...,AZ,BA,BB,BC,...,BZ,...

~~~~~~Your result~~~~~~
Your answer, {BA,CB,AC,DC,ED,BE}, is incorrect. It should have been {AB,BC,CA,CD,DE,EB}

SOLUTION
The **edge set** of a graph is the set of all edges that exist in the graph. However, for a digraph, you need to be careful as direction dictates how each edge is written. For instance, the edge that exists between E and B is EB and not BE.

In this question, alphabetical order still has to be applied, but only to the set of edges, not to the edges themselves. As such, the correct answer is {AB,BC,CA,CD,DE,EB}.

Related material
0 out of 1
You wrote every edge in the wrong direction. Switch the letters in each pair to obtain the correct answer.

Figure 7.3.3: Feedback on the edge set, RWI+check.

7.3.4. Technical content

The coding below is shown on the example presented in Figure 7.3.3 and this is as follows:

- Generation of the correct answer

Correct%QUESTION.NUMBER% = "";

for (ivertex = 1; ivertex <=n; ivertex++){

for (jvertex = 1; jvertex <=n; jvertex++){

if(A[ivertex][jvertex] != 0){Correct%QUESTION.NUMBER% += alphabet(ivertex-1,1) +alphabet(jvertex-1,1)+","}}} *Correct%QUESTION.NUMBER%* is a correct answer constructer using the double 'for loop'. As described in Section 3.1.4 each of the 'for loops' starts at 1 representing the first vertex, increases by 1 each time the script is run, and runs up to *n* (the last vertex). The code to be executed involves the *if* statement which, on the other hand, executes the code only if *A[ivertex][jvertex] != 0* (there is an edge connecting two vertices). Each time this is true, the edge with a comma are added to form the correct answer.

strlength = Correct%QUESTION.NUMBER%.length;

Correct%QUESTION.NUMBER%=
Correct%QUESTION.NUMBER%.substring(0,strlength-1);

There is a comma after the last edge. To make the correct answer of the right formal it has to be deleted. Firstly, the *length* property is used to return the length of a string holding the correct answer. Then, the *substring()* method extracts the characters from that string from the first to the one by last element (not including the last element, that is the comma).

7.4. Spanning trees and minimum spanning trees

A number of definitions are necessary to define the spanning tree. A walk is 'an alternating sequence of vertices and edges of a graph' (Rosen, 2007, p. 622). "A circuit is a closed walk that contains at least one edge and does not contain a repeated edge" (Epp, 2010, p. 644). "A tree is a connected undirected graph with no simple circuits" (Rosen, 2007, p. 683). Bearing in mind these definitions, "a spanning tree for a graph G is a subgraph of G that contains every vertex of G and is a tree" (Epp, 2010, p. 702). A minimum spanning tree is a spanning tree for a connected weighted graph with the least weight of all the spanning trees for the graph.

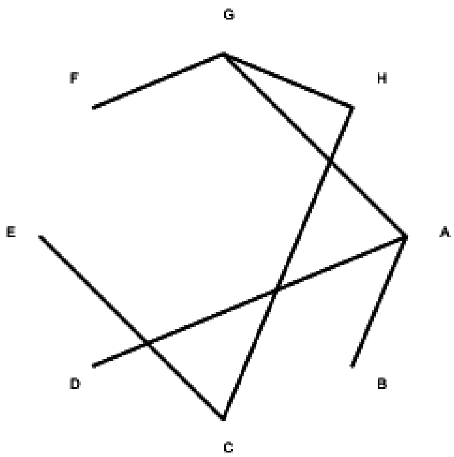
The database comprises of 17 questions of which 4 are of multiple choice type, 1 of numerical input type, and 12 of word input type. These questions are respectively on spanning trees, weight of the minimum spanning tree and on finding the minimum spanning tree using algorithms. Furthermore, questions in each of these groups are of different style.

An example of a question on the spanning tree is presented in Figure 7.4 below.

In a study at University of Western Australia, 8 university student services are being compared and similarities between pairs of services are noted by creating edges and displaying them in the following adjacency matrix.

	A	B	C	D	E	F	G	H
A	0	1	1	1	0	1	1	1
B	1	0	0	1	1	1	1	0
C	1	0	0	0	1	0	1	1
D	1	1	0	0	0	0	1	1
E	0	1	1	0	0	0	1	0
F	1	1	0	0	0	0	1	1
G	1	1	1	1	1	1	0	1
H	1	0	1	1	0	1	1	0

The organizer of this activity wants to show that all services can be connected by a spanning tree. Which, if any, of the following graphs represents this?



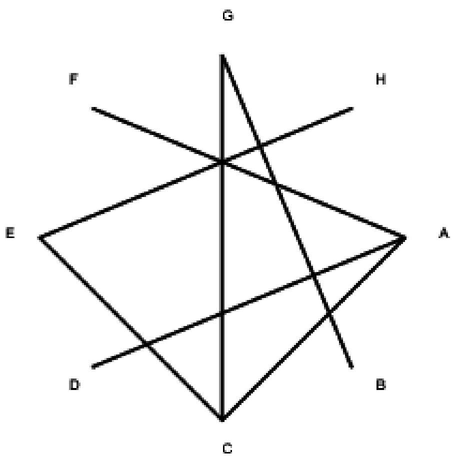


Figure 7.4: Part of the question on the spanning tree, MC.

7.4.1. Assumed and tested skills

It is assumed that students know the definitions of graph, vertex, edge, loop, adjacency matrix (all described in the Section 7.1.1), cycle and weighted graph (described in the Section 7.2.1). Students are also expected to perform simple addition necessary to find the weight of the spanning tree.

Students are tested on their ability on finding the spanning tree of a graph in either the adjacency matrix or graphical representation when it is given either in one these forms. When they are given a network they are tested on finding its minimum spanning tree. The designed questions allow direct testing of the knowledge of two algorithms: Prim’s and Kruskal’s. This is done by either asking them for the total weight, the edge added at indicated step, or the whole minimum spanning tree.

7.4.2. Objective questions

Multiple choice type questions

Students are presented with a graph given either graphically or by the adjacency matrix and asked to find a spanning tree (Figure 7.4). This has to be done by choosing one out of six possible answers. Next to the ‘None of these!’ and ‘I don’t know!’ options there are four graphs given either graphically or by the adjacency matrix presented as possible answers. The displayed graphs are either a correct spanning tree, or distractors. Distractors are formed by adjustments to the correct answer: edge added to form a cycle, edge removed from one vertex, one edge in a spanning tree is not in the original graph or the spanning tree is cut into two smaller trees.

Numerical input type question

A graphical representation of a network is given and the question asks for the weight of a minimum spanning tree. There are no instructions given on what method should be used. A number should be typed in in the input box.

Word input type questions

Students are presented with networks given graphically. All questions require typing in a string of characters. The instructions on the format of the answer depend on the question since groups of questions request different information of a minimum spanning tree.

Different styles

A number of groups of questions can be distinguished depending on what has to be found. Questions in these groups are clones of each other but always differ in some extent. Different presentation of graphs (graphically or by the adjacency matrix) allows differentiation of the questions while the same skill is tested. Within questions on the minimum spanning tree the first difference is with the method of finding the answer, i.e. using Prim's or Kruskal's algorithm. Then questions differ also by the size of the graphs, i.e. some questions involve graphs of 5 to 6 vertices, whereas others 7 vertices (note the coding can cope with any size of graph but 7 vertices is deemed enough to decide whether or not a student can apply the required algorithm). Furthermore, these questions differ by what has to be found by the students. This includes finding the minimum spanning tree, an edge added at specified step, deciding whether the given edge was added, rejected and at what step, or not considered.

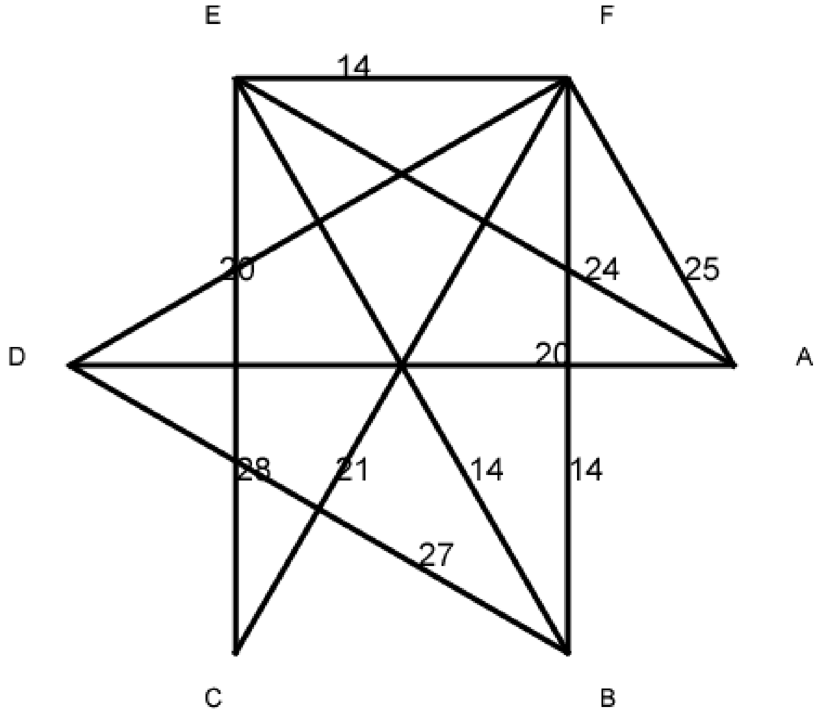
Random parameters

Graphs, networks and adjacency matrices are randomly chosen each time the questions are displayed. Where applicable, the randomisation includes the different number of vertices, different degrees of them, as well as positioning and quantity of isolated vertices. Within questions on spanning trees the displayed question is dressed in one of four available scenarios. Some of them involve random selection of universities' names or countries. Within questions on minimum spanning trees involving algorithms, the number representing the step is randomly selected, as well as the edge that needs to be defined (stating the step at which it was accepted/rejected or not considered).

7.4.3. Feedback

The explanation of the solution (see Figure 7.4.3) starts with the definitions of the minimum spanning tree and the Kruskal's algorithm. From there follow the step-by-step workings of the algorithm.

Find the minimum spanning tree of the following graph using Kruskal's algorithm.



In case you cannot read the weights on the graph, please use the Network Matrix below.

	A	B	C	D	E	F
A	---	---	---	20	24	25
B	---	---	---	27	14	14
C	---	---	---	---	28	21
D	20	27	---	---	---	20
E	24	14	28	---	---	14
F	25	14	21	20	14	---

Input your answer in the box below as an ordered sequence in upper case, with the first edge added by Kruskal's algorithm inputted first, then the second etc. Each edge should be separated by a comma.

~~~~~~Your result~~~~~~

Your answer, **AB,DC,CE,AE,EF**, should have been **BE,BF,AD,DF,CF**.

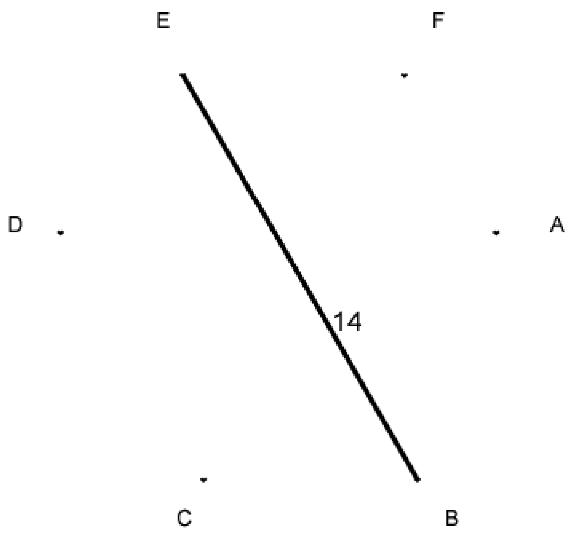
For a connected and undirected network, a **minimum spanning tree** is a connected subgraph of minimum total weight incorporating every vertex of the network.

Kruskal's algorithm: sort all the edges into ascending order of weight. After that, select the edge of least weight to start the tree. Then, consider the next edge of least weight and: if it would form a cycle with the edges already selected, reject it; if it does not form a cycle, consider each in turn. The algorithm terminates when all vertices are connected into one tree.

When finding the minimum spanning tree you should have been connecting edges in the following order:

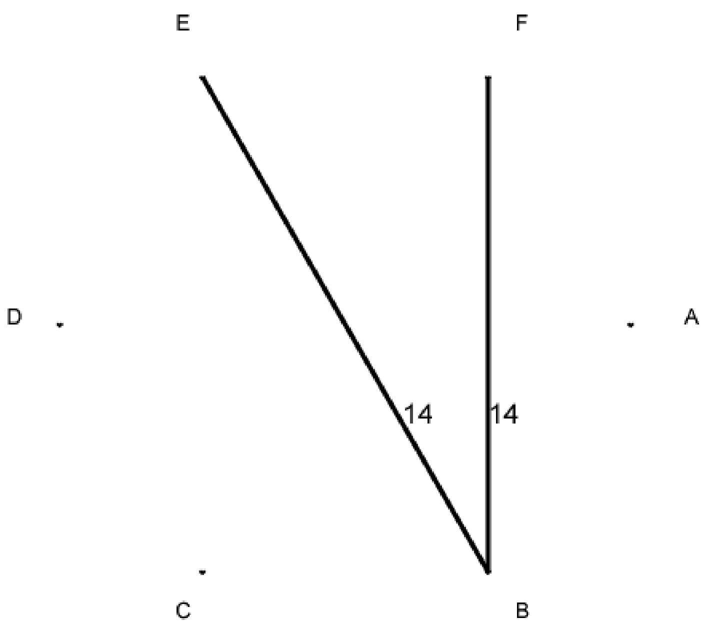
STEP 1

1st edge added i.e. BE



STEP 2

2nd edge added i.e. BF

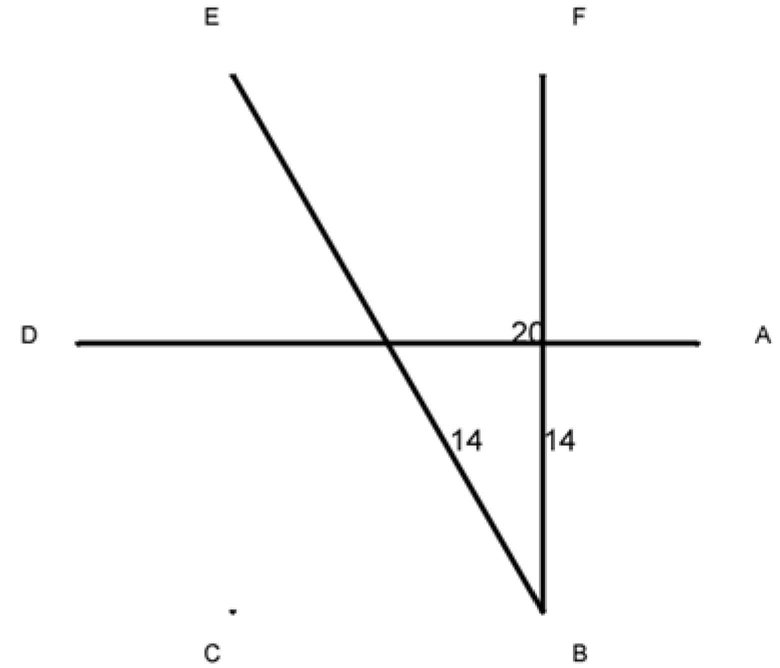


STEP 3

The next edge of least weight is EF however it would form a cycle and therefore is rejected.
~~~~~~

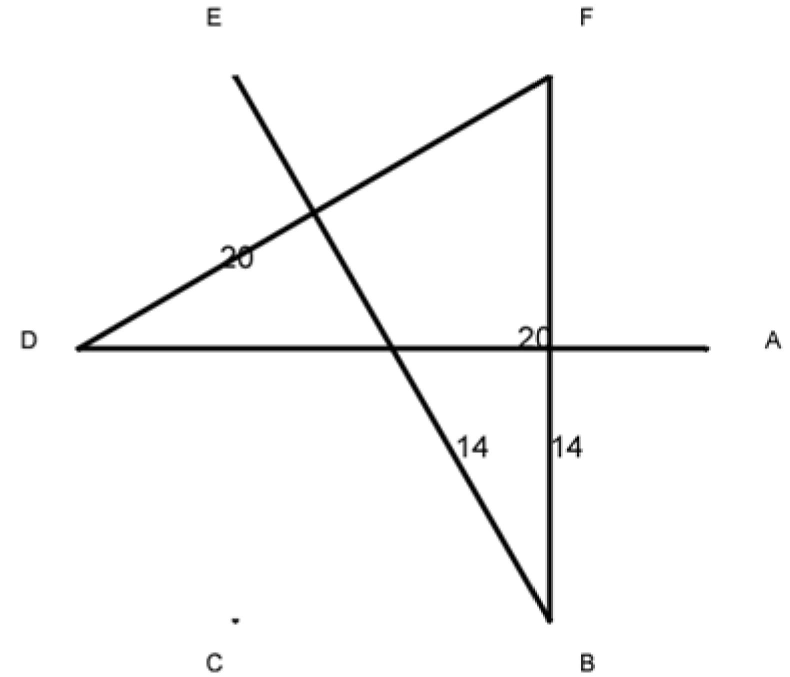
STEP 4

3rd edge added i.e. AD



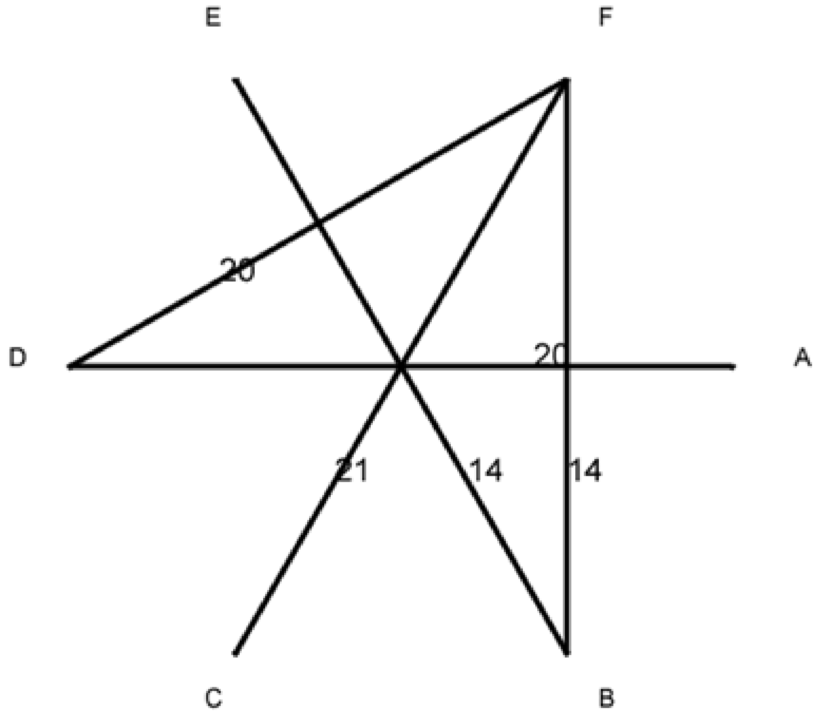
STEP 5

4th edge added i.e. DF



STEP 6

5th edge added i.e. CF



STOPPING CONDITION

Since all the vertices are now connected the minimum spanning tree has already been formed. Hence, the algorithm stops here.

Connecting edges in the indicated order should form a Minimum Spanning Tree:
BE,BF,AD,DF,CF.

Figure 7.4.3: Feedback of the question on the minimum spanning tree using Kruskal's algorithm, WI.

The steps include the information such as the added edge or rejected edge due to formation of the cycle which is prohibited. In contrast to the Prim's algorithm non-connectivity of edges while building the minimum spanning tree is allowed. The coding for the questions also detects the stopping point of the algorithm as all the vertices have been connected and this information is also given to the students. Note that the requirement for inputting edges in correct order is designed to ensure students are actually applying the algorithm correctly.

7.4.4. Technical content

The coding below is shown on the example presented in Figure 7.3.3 and this is as follows:

- Generation of the correct answer

The ‘do/while loop’ loops through the code until the specified condition is true. Within the coding of the question three such loops are used and therefore the code (not showed here) is finally executed when all three conditions are true. The code within the loop is generating the matrix used for the graph to be displayed. It is used to control the size of the random graph.

The *seergedfotsil* is an array holding the increasingly ordered list of degrees. The smallest degree has to be at least 1 and it is the only one since the second is coded to be at least 2. Since there are *n* vertices and arrays start at index zero, the last degree in an array is at the index *n-1.* The condition to be met states that the highest degree can take the maximum value of 6.

The specified conditions prevent the graph to be disconnected or difficult to read due to the high number of edges.

- Generation of the minimum spanning tree (coding and description of the coding in the Appendix E)

The first part of the coding involves function searching through edges in a non-decreasing order and returning them in an array holding: start vertex, end vertex, and the weight of associated with these points edge.

Next, three arrays are created to store information on edges forming the minimum spanning tree, edges forming a cycle, and informing on the role of the considered edge. The allocation of each of these information in the right order is possible thanks to the *push()* method adding new elements at the end of an existing array. A relabeling process of vertices is also present. For this purpose vertices A, B, C,… are renamed by 1, 2, 3… respectively. Then, when the cheapest edge is found, the higher number representing one of the vertices takes the value of the smaller one. This takes place when the edge is to be added to the minimum spanning tree. If at any point, due to the relabeling, the chosen edge has the end vertices of the same value it means that the cycle would be formed and no relabeling takes place.

8. Analysis of the CAA questionnaire

As part of a related study and to establish the questionnaire's validity for use with FoIT students, a CAA questionnaire (Appendix B) was aimed at 259 first-year Mathematics for Economics and Finance students and its findings were described in the 'Proceedings of the 10th International Conference of Technology in Mathematics Teaching (ICTMT10)' (Greenhow and Zaczek, 2011). It was distributed in 2010 and consisted of 12 open and closed questions assessing students' perception on the CAA and 6 general questions about them. The results showed that 79% of students use the CAA for both assessment and as learning material (21% for assessment only). Moreover, students were generally more successful in the questions providing full feedback compared to those lacking it. Moreover, the majority of students use all 5 allowed attempts to get the best possible mark, most students stated that they would use the assessments, even if not compulsory. One could conclude from there that students think positively about the CAA. This can also be confirmed by the results on enjoyment of (maths : CAA): Hate it (1%:0%), Don't enjoy (5%:4%), Neutral (36%:56%), Enjoy (45%:29%), Love it (14%:11%). This might imply that when doing the CAA they regard themselves as doing mathematics itself and see no distinction despite a difference in medium (paper/CAA). The cross-tabulations showed that the more enthusiastic students are about the subject the more they use the feedback as a learning resource. Another interesting cross-tabulation shows that, as suspected, weaker students benefitted relatively more from the CAA: 19:79 'good' students (categorised as having AS grade A or A level grade C or above) use the CAA only for marks (perhaps they do not need the feedback) in contrast to weak students (not in the above category) where this ratio is 6:39.

This questionnaire was distributed during the lecture and the return rate reached 65% of the whole cohort signed for the module. Moreover, almost 90% of the respondents had Mathematics A level at grade C or above. In comparison, the same questionnaire was given to the 102 FoIT students during the Discrete and Decision Mathematics lecture in 2011. It was completed by only 30% and only a few students claimed to have the above mentioned grades. This, as well as the failure rate (grades E, 30%-39% and F, 0 – 29%) of the modules (38% for FoIT (Brunel University 2011) and 25% for Economics students (Greenhow and Zaczek, 2011)) could indicate that FoIT students were generally less motivated and engaged with their education. Due to these negative figures the comparison with Economics attempt to quantify to some extent whether or not weak students feel more or less negative about using CAA. Moreover, I would recommend running the questionnaire with the much stronger group, e.g. Electrical Engineering, but it will be addressed in Recommendations.

8.1. Single questions

Twelve main questions from the questionnaire delivered to FoIT students are on CAA. On top of them, students were asked how do they feel about maths and how do they feel about it in comparison to other modules. Maths was defined as ‘harder’ or ‘very much harder’ by the total of 51% of respondents. This result was different from the one obtained from the Economics students, as only 33% of them put maths into either of these two groups. These students take different maths modules and their maths qualifications also differ, but their enjoyment of the subject is comparable and rather positive, see Figure 8.2a. Moreover, the similarities are present in their responses to most of the questionnaires’ questions.

It happened that results of how FoIT students engage with the CAA is broadly the same as in the case of Economics students. For the majority (79%) of students it is not only an assessment but also a learning tool leaving 21% of them thinking about it as an assessment only. 82% use all 5 available attempts trying to get the best possible result. Most students stated that they would use the assessments, even if not compulsory: as much as now (18%), once or twice (32%), only for the feedback (29%) and not at all (21%) (see Figure 8.1).

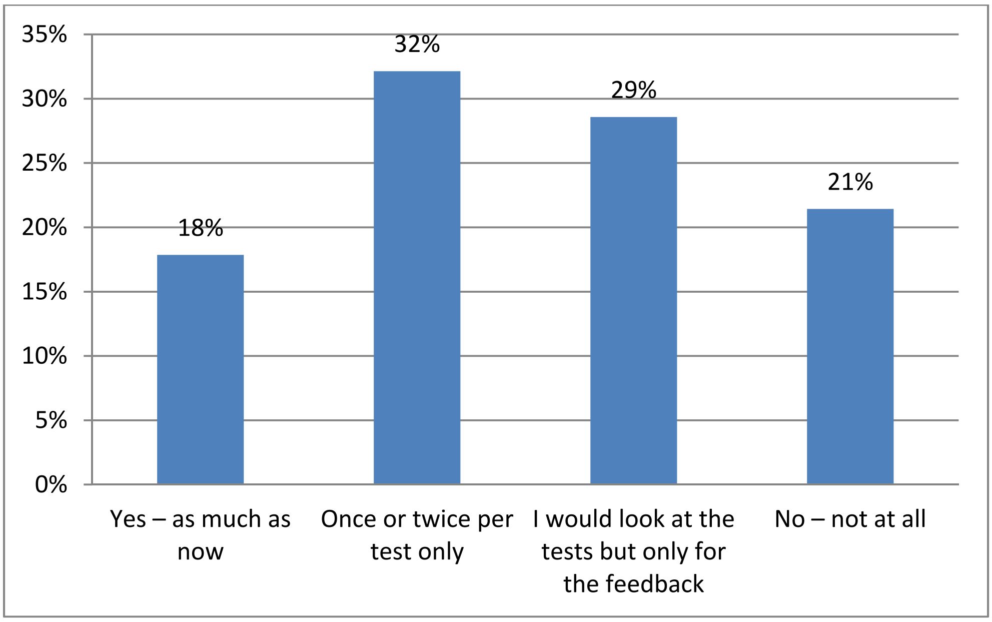


Figure 8.1: FoIT students’ responses to the question ‘Would you use tests as the learning source if they were not compulsory and the results not counted towards your final grade? Be honest!’

This shows fairly equal distribution but the proportion of students indicating the last option in comparison to Economics students is higher and may once more prove less engagement with the subject. The expectations of FoIT students, towards their grades are smaller, as only 68% of students (85% Economics) aim to achieve a mark of at least 80% which may indicate some pride in their progression and a sense of achievement rather than simply marks accrual. Moreover, 72% plan to use the CAA as part of their revision even though marks will not then count. It is a majority but less than Economics students by 12%. The questionnaire results indicate that substantially more students help others than are helped by others.

The questionnaire also included three open questions. They were asking what students like most and least about CAA but there was also a chance to share ideas on how the CAA could be improved or changed. Collected opinions show that students appreciate the features CAA brings. The positives include:

- ‘They give you 5 attempts and also feedback so you can get a better result in the next attempt’
- ‘it helps to understand all the topics covered in lec’s’
- ‘they count towards your overall grade’
- ‘they allow students to work together’
- ‘no time limit’
- ‘different style of questions’

On the other hand, students are aware of some technical downsides of CAA. These are ‘the waiting after submitting an answer’ and ‘the grey screen. It’s dull’. However, students can change background/font colours but possibly they did not see this feature available in the top right corner of the screen when solving tests. Furthermore, in students’ opinion and what they do not like is that there are

- ‘too many questions’,
- ‘there is only 5 attempts’,
- ‘one little mistake reduces your percentage’,
- ‘very very hard questions’
- ‘questions are very predictable’
- ‘some errors’

Most of the recommendations and ideas for improvement were a consequence of what students did not favour e.g. ‘less questions’, ‘be easier’, ‘more attempts’, etc. However, the more constructive feedback like:

- ‘better related material for every question’

- ‘making some of the questions more clear’
- ‘make it work so I could do them from home’

resulted in: questions being revisited by people designing them and rewording where necessary, website links to related materials have been updated by new ones and not working ones being deleted. Making the software being available from home was under the development and is now accessible through a variety of devices supported by different operating systems. This is possible on Macs and PCs (any browsers except Safari on Macs) using *maths e.g.* (Maths E.G. n.d.) package. However, work takes place in order to resolve this problem.

8.2. Cross-tabulations of questions

Researchers are often interested in differences in different aspects in relation to gender. The claim that men are more keen on using the computers is commonly said by people. The study that tested the relation between gender and attitude towards the computers was carried out and shows that in most countries this is the case (Leino, n.d.). Similar results were obtained from the Economics and FoIT students. In both groups men are more likely to show the strongest positive approach (‘Love it’ response) when asked about their feelings about the use of CAA in the mathematics module (see Figure 8.2a).

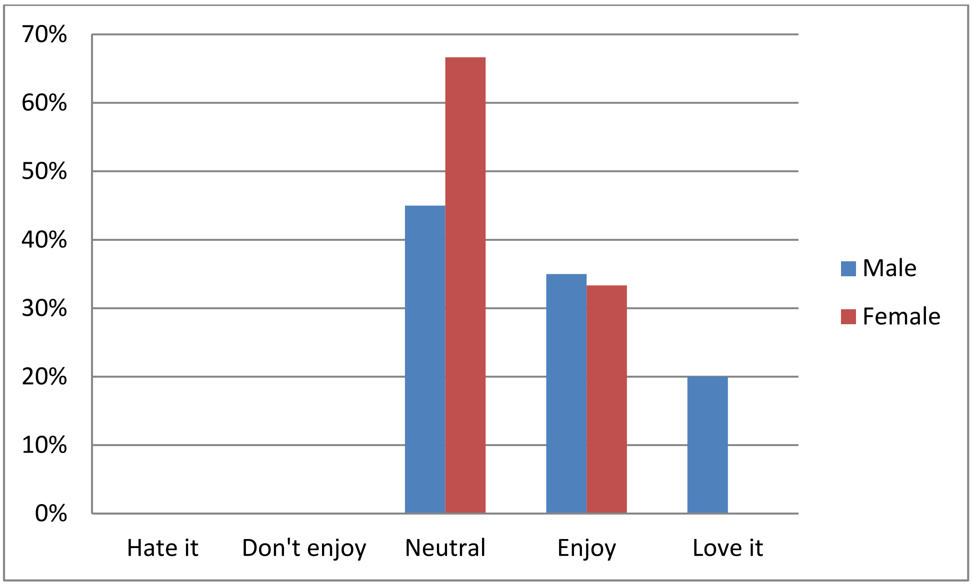


Figure 8.2a: Cross-tabulation of FoIT students’ responses: How do you feel about the use of CAA in the maths module? by Gender.

Students' answers to the questions about their feeling towards mathematics and using CAA were both mainly positive, with maths being favoured slightly more often. Both these aspects were put against the way students treat CAA (assessment only/assessment and learning material). The enjoyment of CAA compared to the enjoyment of maths is higher within the students seeing CAA as a learning material and smaller when considering those who only treat it as an assessment. A reasonable inference is that good quality feedback affects the perception of CAA although we cannot yet claim a causal link. One more factor to look at could be the willingness of using the software for the revision before the exam. It has been tested and the results show that students who see benefits from the CAA beyond being only an assessment tool are more likely to use it for their revision (see Figure 8.2b).

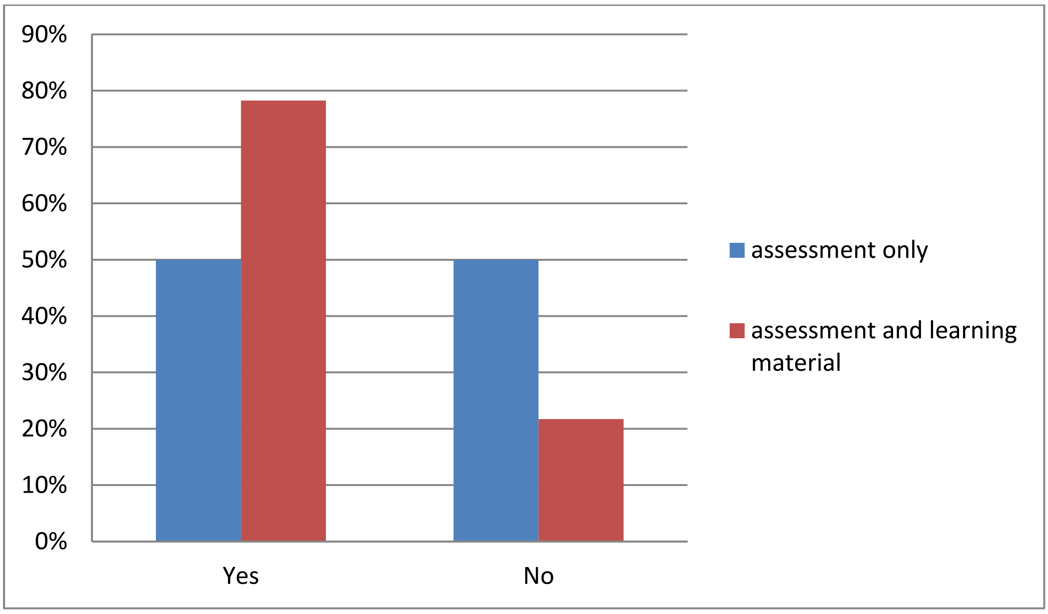


Figure 8.2b: Cross-tabulation of FoIT students' responses: Are you planning to use tests for your revision before the exam? by How do you treat CAA tests?

Many more cross-tabulations could be produced and a few more beyond what is presented above were generated. However, it is not easy to analyse the effects that one result could have on another. Most cross-tabulations produced show similar preferences of FoIT students to those of the Economics students. The link between using the CAA if it was not compulsory and the way it is perceived by students was also considered and the match between two groups of students was found in 50%. The first and third set of bars in Figure 8.2c showed the same order of preference according to how CAA was regarded. Describing the results of FoIT students, those treating tests as an assessment only are more likely than those who see it also as a learning tool to do tests once or twice per test or only look at them for the feedback. One could say that the second comparison is surprising as the answers are mutually exclusive. It seems contradictory that some students say they do not use the tests as learning material but then say they would use them for feedback.

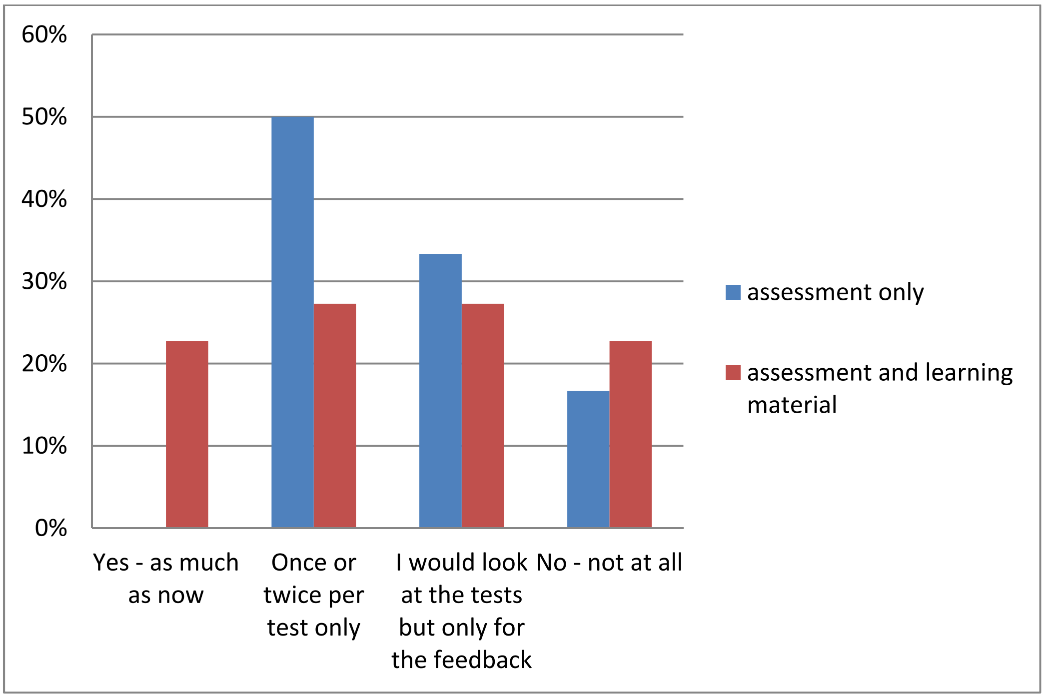


Figure 8.2c: Cross-tabulation of FoIT students' responses: Would you use tests as the learning source if they were not compulsory and the results not counted towards your final grade? by How do you treat CAA tests?

Furthermore, students who see CAA as an assessment and learning material are less likely to access the tests if they were not compulsory. These results bring the question of how accurate participants' perceptions are.

An interesting cross-tabulation could have been to assess who benefits more from the CAA (as done for the students taking Economics). However, not many students provided the information on their maths qualifications and since the questionnaire was anonymous it is impossible to match the data of prior qualifications with the responses from the questionnaire. On the other hand, all the students could be considered as weak since in most cases lack of maths qualification was a reason for taking the course before proceeding to the first year of the undergraduate study. The questions on how CAA is treated, whether it would be used if not compulsory, for revision or simply an enjoyment of CAA were answered by the majority of students in a way indirectly suggesting benefits for weak students.

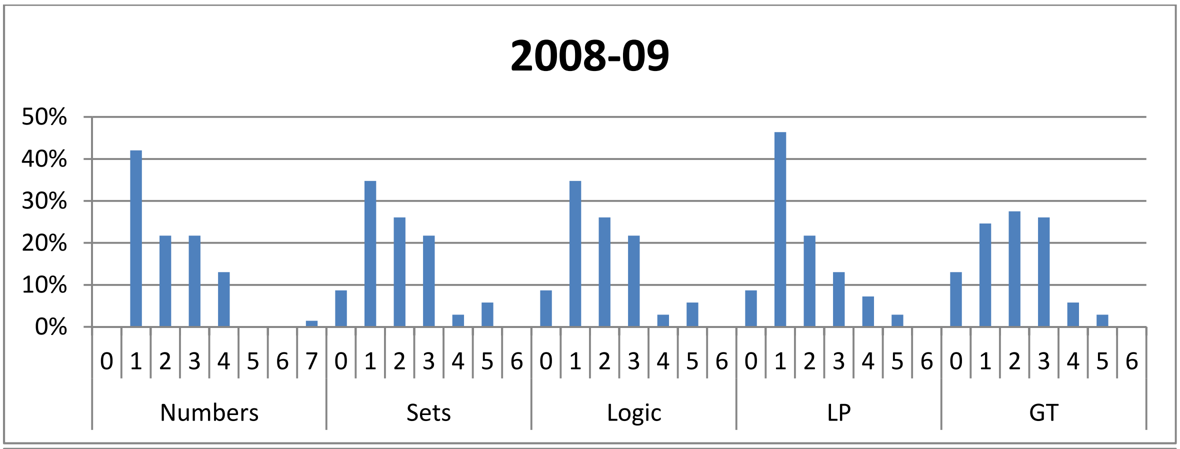
9. Analysis of answer files

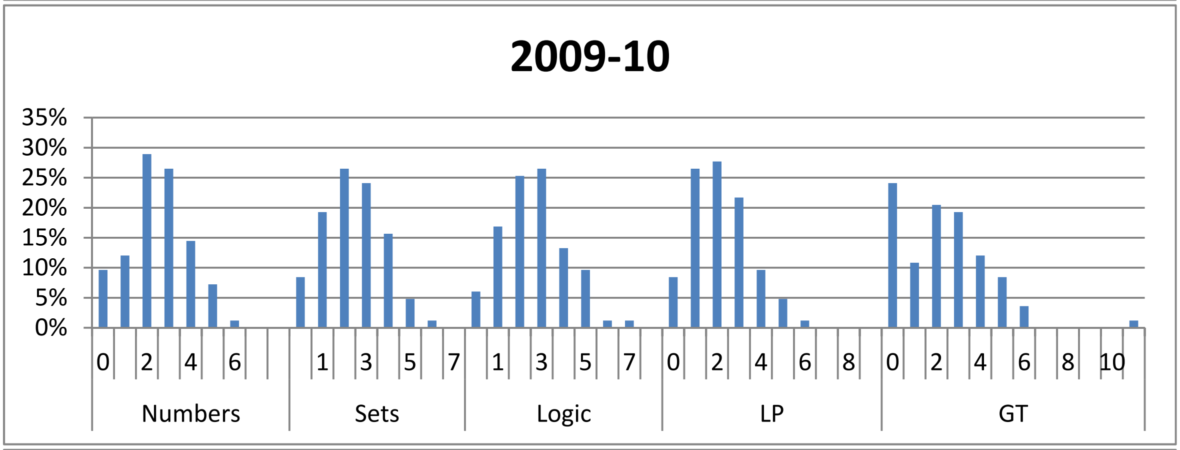
Apart from giving marks to students, the CAA system facilitates easy analysis of answer files. This gives teaching staff a great deal of hard information that would otherwise remain largely anecdotal or based on the experience of the lecturer. Three useful statistics are discussed below, with illustrative examples. These are student engagement in the tests throughout five topics and five years, question facility and question discrimination. These help to inform subsequent teaching and improvement of questions, as discussed in the Recommendations, Section 11.5.

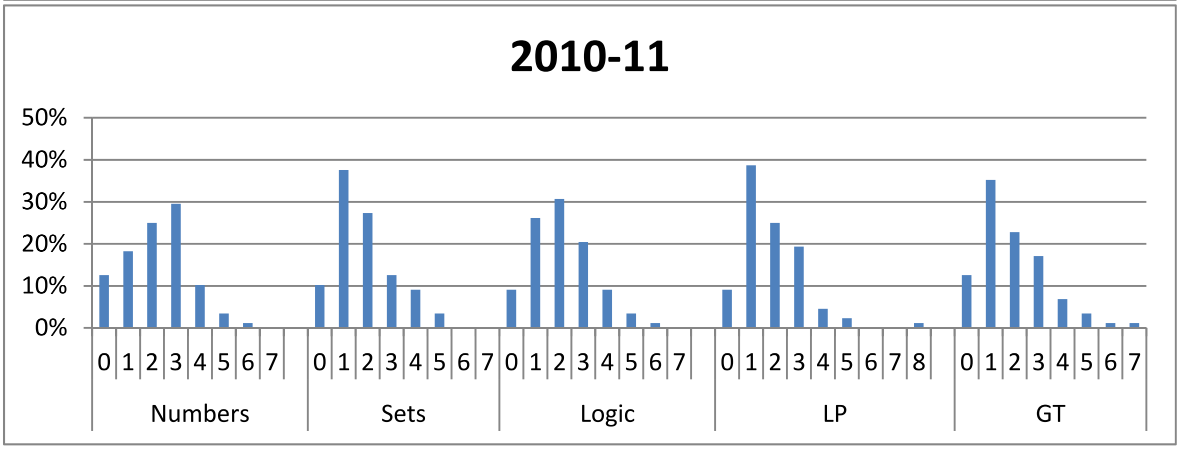
9.1. Engagement

As a measure of student engagement and to understand the possible knock-on effect on students' exam performances (see Chapter 10) we present data here on the number of times students attempted each of the five tests over five academic years. The multiple attempts at tests confirms what was found in Chapter 8, that the majority of students perceive CAA tests as a learning resource and appreciate the available and instant feedback.

Although only the best of five possible attempts counts, some students decided to take tests more than this. Figure 9.1 shows that in the 2009-10 academic year one student attempted the graph theory test 11 times, stopping when the score was 83%. This however is an extreme single case. The bar charts below show the distribution of the number of attempts (completed tests) for each of the five topics, in each academic year.


2008-09
50%
40%
30%
20%
10%
0%
Numbers
Sets
Logic
LP
GT


2009-10
35%
30%
25%
20%
15%
10%
5%
0%
Numbers
Sets
Logic
LP
GT


2010-11
50%
40%
30%
20%
10%
0%
Numbers
Sets
Logic
LP
GT

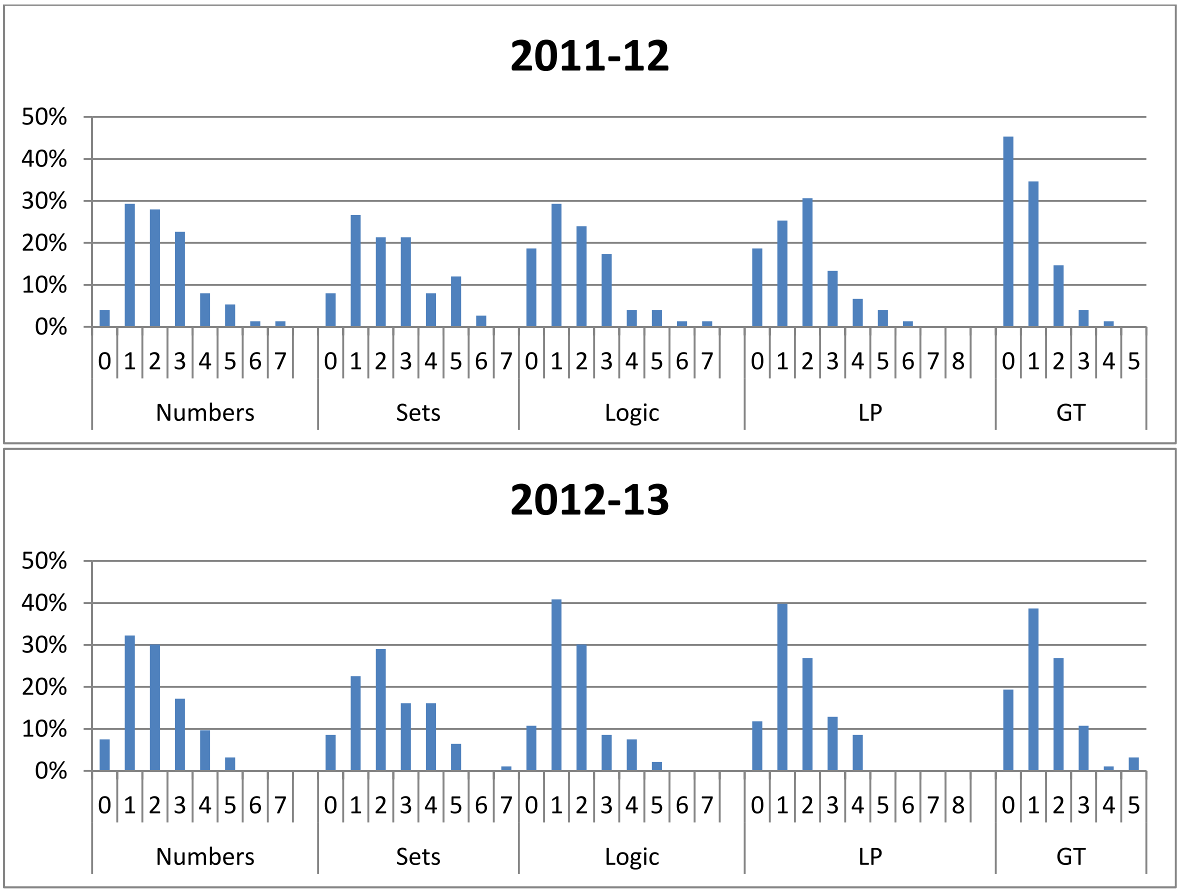


Figure 9.1: The distribution of the number of attempts (completed tests) for each of the five topics in each academic year. The vertical axis is the percentage of students taking the tests specified number of times in each topic. Key: LP = Linear programming, GT = Graph theory.

Reading the mode from the bar charts for each group is straightforward. In most (14) cases it is 1, 2 was the mode in seven cases, with 0 and 3 being the mode twice. These results vary between years and topics, and therefore no direct conclusion can be drawn showing a trend. On the other hand, there is a trend for the percentage of students not completing the test at all (0 attempts) and this could be because they have dropped-out of all their studies but this is unknown. It rises as one progresses through the topics. This shows that some students completely disengage from the later topics, especially graph theory (GT).

The mean number of completed tests is shown in Table 9.1a.

Table 9.1a: Mean number of completed tests in each year, for each topic and overall.

	Numbers	Sets	Logic	LP	GT	Overall
2008-09	2.2	2.0	2.0	1.8	2.0	2.0
2009-10	2.5	2.4	2.7	2.2	2.4	2.4
2010-11	2.3	1.8	2.1	1.9	2.0	2.0
2011-12	2.3	2.4	1.8	1.8	0.8	1.8
2012-13	2.0	2.3	1.7	1.7	1.5	1.8

One could say that there was a decrease in the interest in the CAA with average number of completed tests in 2011-12 and 2012-13 equal to 1.8. This can be explained by normalising the data (Table 9.1b) according to the number of questions attempted in each test i.e calculating the mean number of questions attempted. The tests not only have a varying number of questions between the topics, but also increased each year as further questions became available. Normalised data allows the better comparison as the impact of the variable number of questions on the results is then taken into account.

Table 9.1b: Normalised mean number of questions in completed tests in each year, for each topic and overall.

	Numbers	Sets	Logic	LP	GT	Overall
2008-09	15.1	3.9	11.7	8.8	12.1	10.3
2009-10	17.6	14.6	16.1	10.8	14.6	14.7
2010-11	15.8	11.0	12.6	9.5	15.6	12.9
2011-12	16.0	14.3	12.6	8.9	7.2	11.8
2012-13	17.9	21.0	15.1	8.3	13.1	15.1

Looking at the normalised data some patterns can be seen. There is a significant decrease in the number of questions students completed in the linear programming topic throughout five years, but an increase for the graph theory topic. This can be explained, at least partially, for the 2012-13 year when a compulsory exam question on graph theory was introduced, worth 40 % and spanning most of the major skills taught for this topic. Thus engagement with linear programming may have decreased as students strategically chose to avoid that topic on the exam (they have to do the compulsory graph theory question plus questions on 3 out of the 4 other topics). An exception year was 2011-12 when engagement in graph theory decreased even further (this partly triggered the compulsory graph theory question in the following year). Analysis of the data from years 2008-2011 confirmed speculations that students' commitment to the subject/university drops with time. Comparison of the engagement in the numbers test and the one on the graph theory shows that every year fewer questions were answered at the end of the year than at the beginning.

9.2. Facility

For the dichotomous marking scheme we generally use, question facility is simply the mean mark. It gives the percentage of the participants that answered the question correctly. If the statistic is high it means that the question is easy and the majority answer it correctly. This on the other hand tells teachers on which topics they should spend more time. If the value is very low it can mean that the question is "too hard, confusing or misleading, problem with content or instruction" (Peterson, 2013, p. 14). Assuming that the question is free of mistakes, the facility can be influenced by the question content, but also by the question type.

The CAA answer files comprise five years. The entry requirements for the course did not change throughout the time and also the engagement with CAA proved to be comparable (see Section 9.1). Based on this information it can be assumed that the ability and attitude of the cohort does not change over time and therefore the data from all years has been combined. As a result, conclusions presented below are drawn from a very large dataset and should therefore be more informative. This is because any outlying results specific to one year are smoothed by the results from other years. The tables of combined years are included in this chapter, whereas those on each topic for each year are in Appendix C. The tables consist of five columns: question description, topic, times answered, mean score and correlation (considered in Section 9.3). The entries in the third column (times answered) differ in values and there are two possible reasons for that. Firstly, some questions were designed later than others and therefore answered in different number of years. Secondly, the topics include different numbers of questions and when drawn at random for the tests had different chances of being selected.

9.2.1. Numbers

Questions on the Numbers topic are of the NI type or WI type when the answer is a list of numbers and/or characters. Therefore, guessing is limited and there are no clues as in the case of, for example, MC type questions. Questions that students find relatively easy include those on conversion between decimal and scientific notation of numbers (Figure 3.4), prime factors (Figure 3.1, Figure 3.1.1), highest common factor (Figure 3.1.2a), and lowest common multiple (Figure 3.1.2b). Questions on non-decimal arithmetic asking to find the result of addition (Figure 3.3), subtraction, multiplication, division of two numbers of one of three available bases, or finding the representation of a number in different base (Figure 3.3.3a, Figure 3.3.3b) resulted in a mean score above 0.5 i.e. most of the time correct answers were submitted. Questions on modular arithmetic (Figure 3.2) and significant figures caused more problems. The most difficult questions from five years are those on non-decimal arithmetic involving ones/sevens/Fs complement for subtraction. Table 9.2.1 contains the combined statistics from five years on numbers, organised from the easiest to the most difficult question. (The data is given in Appendix C as a spreadsheet allowing the reader to sort the data by other criteria, such as topic or correlation.)

Table 9.2.1: Combined questions from five years on numbers, organised by their facility/mean score.

Question description	Topic	Times answered	Mean score	Correlation
scientific number A 10^B, B lt -2 to decimal; RNI	Numbers\Scientific notation	49	0.980	0.310
scientific number A 10^B, B gt 2 to decimal; RNI	Numbers\Scientific notation	55	0.909	0.355
binary to decimal; NI	Numbers\Non-decimal arithmetic\Number bases	133	0.872	0.266
multiplication with table; NI	Numbers\Non-decimal arithmetic\Octal arithmetic	19	0.842	0.583
decimal to scientific number A 10^B, B lt -2; 2NI	Numbers\Scientific notation	33	0.818	0.641
division; NI	Numbers\Non-decimal arithmetic\Hexadecimal arithmetic	171	0.813	0.435
addition; NI	Numbers\Non-decimal arithmetic\Octal arithmetic	175	0.811	0.578
decimal to scientific number A 10^B, B gt 2; 2NI	Numbers\Scientific notation	39	0.795	0.148
addition with table; NI	Numbers\Non-decimal arithmetic\Octal arithmetic	29	0.793	0.716
factor primes multiplied; WI	Numbers\Prime factorisation	271	0.786	0.533
subtraction; NI	Numbers\Non-decimal arithmetic\Octal arithmetic	179	0.782	0.548

addition; NI	Numbers\Non-decimal arithmetic\Hexadecimal arithmetic	162	0.778	0.576
addition; NI	Numbers\Non-decimal arithmetic\Binary arithmetic	193	0.772	0.479
division; NI	Numbers\Non-decimal arithmetic\Octal arithmetic	172	0.767	0.568
HCF primes1*primes2 arrays; NI	Numbers\Prime factorisation	259	0.764	0.547
binary to octal; NI	Numbers\Non-decimal arithmetic\Number bases	129	0.752	0.419
division; NI	Numbers\Non-decimal arithmetic\Binary arithmetic	164	0.750	0.469
binary to hexadecimal; NI	Numbers\Non-decimal arithmetic\Number bases	127	0.748	0.321
find primes of N, prime factor tree inc; XNI	Numbers\Prime factorisation	171	0.745	0.590
hexadecimal to decimal; NI	Numbers\Non-decimal arithmetic\Number bases	136	0.743	0.453
addition with table; NI	Numbers\Non-decimal arithmetic\Hexadecimal arithmetic	23	0.739	0.510
multiplication; NI	Numbers\Non-decimal arithmetic\Octal arithmetic	170	0.724	0.604
LCM primes1*primes2 arrays; NI	Numbers\Prime factorisation	270	0.722	0.462
subtraction; NI	Numbers\Non-decimal arithmetic\Hexadecimal arithmetic	179	0.698	0.521
multiplication; NI	Numbers\Non-decimal arithmetic\Binary arithmetic	210	0.695	0.634
decimal to binary; NI	Numbers\Non-decimal arithmetic\Number bases	123	0.667	0.439
number of sig figs; 4SN	Numbers\SigFigs	333	0.667	0.42
decimal to hexadecimal; NI	Numbers\Non-decimal arithmetic\Number bases	107	0.664	0.595
subtraction; NI	Numbers\Non-decimal arithmetic\Binary arithmetic	186	0.624	0.527
a+b+c to sf; RNI	Numbers\SigFigs\random questions	29	0.621	0.549
base 3 to 9 to decimal; NI	Numbers\Non-decimal arithmetic\Number bases	92	0.620	0.614
multiplication; NI	Numbers\Non-decimal arithmetic\Hexadecimal arithmetic	168	0.619	0.515
multiplication with table; NI	Numbers\Non-decimal arithmetic\Hexadecimal arithmetic	27	0.593	0.439
decimal to base 3 to 9; NI	Numbers\Non-decimal arithmetic\Number bases	134	0.575	0.625
scientific number^(p/q); 2NI	Numbers\Scientific notation	194	0.536	0.591
scientific number^n; 2NI	Numbers\Scientific notation	168	0.530	0.481
x mod p = a, y mod p = b, find (x+y) mod p; NI	Modular arithmetic	371	0.507	0.533
division of two scientific numbers; 2NI	Numbers\Scientific notation	221	0.484	0.607
x mod p = a, y mod p = b, find (x-y) mod p; NI	Numbers\Modular arithmetic	360	0.481	0.505

product of two scientific numbers; 2NI	Numbers\Scientific notation	204	0.471	0.470
x mod p = a, y mod p = b, find xy mod p; NI	Numbers\Modular arithmetic	354	0.463	0.597
numbers axb range; 2NI	Numbers\SigFigs\random questions	26	0.462	0.359
square root of a scientific number; 2NI	Numbers\Scientific notation	203	0.458	0.519
axbxc to sf; RNI	Numbers\SigFigs\random questions	33	0.424	0.329
a/(bxc) to sf; RNI	Numbers\SigFigs\random questions	30	0.400	0.397
axb/c to sf; RNI	Numbers\SigFigs\random questions	45	0.378	0.626
SA of tank axbxc to sf; RNI	Numbers\SigFigs\random questions	37	0.378	0.005
check prime list; WI	Numbers\Prime factorisation	271	0.302	0.449
numbers a+b range; 2NI	Numbers\SigFigs\random questions	37	0.270	0.265
numbers a-b range; 2NI	Numbers\SigFigs\random questions	26	0.231	0.581
Fs complement for subtraction; NI	Numbers\Non-decimal arithmetic\Hexadecimal arithmetic	139	0.187	0.537
ones complement for subtraction; NI	Numbers\Non-decimal arithmetic\Binary arithmetic	172	0.180	0.314
sevens complement for subtraction; NI	Numbers\Non-decimal arithmetic\Octal arithmetic	167	0.138	0.446
subtraction a-b b gt a; NI	Numbers\Non-decimal arithmetic\Binary arithmetic	29	0.069	0.174
numbers a/b range; 2NI	Numbers\SigFigs\random questions	23	0.043	0.333

According to the information on the difficulty level of certain topics on numbers it is recommended that more teaching time should be spent on complements, significant figures and possibly on modular arithmetic.

9.2.2. Sets

Table 9.2.2 shows that questions on sets that students find relatively easy include those which test students' knowledge on power sets (Figure 4.4, Figure 4.4.2a and b, Figure 4.4.3). Also questions asking to find union, intersection, difference, complement of a set show a tendency towards high to medium mean marks. Questions involving sets of integers are slightly harder. A similar situation applies to questions that require listing set elements explicitly (Figure 4.1a). They are relatively easy but those involving integers or absolute values cause more problems to students. Furthermore, questions like 'Input the following set explicitly with elements in numerical order: A = $\{x \in \mathbb{Z}^{+} \mid |x| \geq 4\}$' prove to be the most difficult of all questions on sets. Other difficult questions are those of the True/False/Undecidable (TFU) type which require all four inputs judging the truth of statements (on subsets, sets equality, set identities; see Figure 4.2, Figure 4.10) to be correct in order to get a mark. Questions on the counting principle (Figure 4.12.3) are characterised by the lowest mean score (especially with 3 sets). This may be due to the need to translate the given information into mathematics and using knowledge on sets union, intersection, subsets, etc.

Table 9.2.2: Combined questions from five years on sets, organised by their facility/mean score.

Question description	Topic	Times answered	Mean score	Correlation
given A and B, A = B, find (A\B) ; WI	Logic\Sets\Difference	34	0.971	0.276
intersection of two sets_when given sets A, B, C, D explicitly; WI	Logic\Sets\Intersection	58	0.931	0.446
given number of elements in the set find the number of elements in the power set; RNI	Logic\Sets\Power set	47	0.872	0.448
find (AnB) complement when (AnB) = U; WI	Logic\Sets\Complement	53	0.868	0.468
given A and B, A = B, find (B\A) ; WI	Logic\Sets\Difference	30	0.867	0.729
find n'th element of P(B) when it doesn't have that many elements; WI	Logic\Sets\Power set	43	0.860	0.580
find A complement when A = U; WI	Logic\Sets\Complement	58	0.828	0.466
Input set elements explicitly xeN a lt x le b; WI	Logic\Sets\Elements	54	0.815	0.357
given A and B, n(A)=n(B), find (B\A) ; WI	Logic\Sets\Difference	43	0.814	0.536
find A complement; WI	Logic\Sets\Complement	63	0.810	0.424
2 subjects; NI	Logic\Sets\Counting	45	0.800	0.546
given A and B, n(B) le n(A), find (B\A) ; WI	Logic\Sets\Difference	33	0.788	0.624
given A and B, n(A) ge n(B), find (A\B) ; WI	Logic\Sets\Difference	28	0.786	0.442
given finite universal set_ find A complement; WI	Logic\Sets\Complement	228	0.772	0.607
Input set elements explicitly xeN a le x le b; WI	Logic\Sets\Elements	65	0.769	0.604
intersection__Input set elements explicitly xeN x le ge b; WI	Logic\Sets\Intersection	94	0.755	0.624
Input set elements explicitly xeN a lt x lt b; WI	Logic\Sets\Elements	109	0.752	0.625
given A and B, n(A)=n(b), find (A\B) ; WI	Logic\Sets\Difference	32	0.750	0.607
given A and B, n(B) ge n(A), find (B\A) ; WI	Logic\Sets\Difference	36	0.750	0.679
power set of the set containing 3 elements; MC	Logic\Sets\Power set	44	0.750	0.530

Input set elements explicitly xeN x lt b; WI	Logic\Sets	83	0.747	0.482
Input set elements explicitly xeN a le x lt b; WI	Logic\Sets\Elements	67	0.746	0.697
find (AuB) complement when (AuB) = U; WI	Logic\Sets\Complement	59	0.746	0.371
Input set elements explicitly xeN abs(x) lt b; WI	Logic\Sets\Elements	39	0.744	0.686
given universal set, input set elements explicitly a le x le b; WI	Logic\Sets\Elements	35	0.743	0.580
find A complement when A = { }; WI	Logic\Sets\Complement	58	0.741	0.454
find A complement when A != U & { }; WI	Logic\Sets\Complement	53	0.736	0.530
intersection__Input set elements explicitly xeN x lt ge b; WI	Logic\Sets\Intersection	111	0.712	0.615
find (AuB) complement when (AuB) != U ; WI	Logic\Sets\Complement	76	0.711	0.688
union__Input set elements explicitly xeN x lt le b; WI	Logic\Sets\Union	122	0.705	0.537
intersection__Input set elements explicitly xeN x gt ge b; WI	Logic\Sets\Intersection	106	0.698	0.538
given A and B, n(A) le n(B), find (A\B) ; WI	Logic\Sets\Difference	33	0.697	0.764
intersection__Input set elements explicitly xeZ x ge ge b; WI	Logic\Sets\Intersection	112	0.696	0.642
given universal set, input set elements explicitly a lt x lt b; WI	Logic\Sets\Elements	45	0.689	0.603
find n'th element of P(B) when n(B)=3; WI	Logic\Sets\Power set	38	0.684	0.542
Input set elements explicitly xeN x le b; WI	Logic\Sets\Elements	99	0.677	0.567
intersection__Input set elements explicitly xeN x le lt b; WI	Logic\Sets\Intersection	102	0.676	0.514
intersection__Input set elements explicitly xeN x ge ge b; WI	Logic\Sets\Intersection	92	0.663	0.576
find (AuB) complement; WI	Logic\Sets\Complement	50	0.660	0.543
given universal set, input set elements explicitly a le x le b; WI	Logic\Sets\Elements	11	0.636	0.847
Input set elements explicitly xeZ x gt b; WI	Logic\Sets\Elements	53	0.623	0.593
intersection__Input set elements explicitly xeN x le le b; WI	Logic\Sets	142	0.620	0.580
intersection__Input set elements explicitly xeZ x gt ge b; WI	Logic\Sets\Intersection	97	0.619	0.604
2 subjects,none; NI	Logic\Sets\Counting	47	0.617	0.511
union__Input set elements explicitly xeZ x ge ge b; WI	Logic\Sets\Union	107	0.617	0.556
Input set elements explicitly xeZ x ge b; WI	Logic\Sets\Elements	60	0.617	0.750
union__Input set elements explicitly xeN x ge gt b; WI	Logic\Sets\Union	95	0.610	0.658
given the set find the number of elements in the power set; NI	Logic\Sets\Power set	33	0.606	0.804
Input set elements explicitly xeN x ge b; WI	Logic\Sets	103	0.602	0.550
union__Input set elements explicitly xeN x le le b; WI	Logic\Sets	148	0.595	0.715

Input set elements explicitly xeN abs(x) ge b; WI	Logic\Sets\Elements	51	0.588	0.788
find (AnB) complement; WI	Logic\Sets\Complement	114	0.579	0.594
find n'th element of P(B) when n(B)=4; WI	Logic\Sets\Power set	45	0.578	0.709
union__Input set elements explicitly xeN x le ge b; WI	Logic\Sets\Union	104	0.558	0.634
union__Input set elements explicitly xeN x ge ge b; WI	Logic\Sets	139	0.554	0.665
Input set elements explicitly xeN x gt b; WI	Logic\Sets\Elements	94	0.553	0.700
3 subjects_all 3; NI	Logic\Sets\Counting	46	0.522	0.566
two sets relationship_A subset of B; 4TFUSP	Logic\Sets\Subsets	98	0.520	0.644
two sets relationship; 4TFUSP	Logic\Sets\Subsets	85	0.518	0.407
union__Input set elements explicitly xeZ x ge gt b; WI	Logic\Sets\Union	103	0.515	0.657
two sets relationship_B subset of A; 4TFUSP	Logic\Sets\Subsets	78	0.513	0.617
find (AnB) complement when (AnB) != U & { }; WI	Logic\Sets\Complement	59	0.508	0.660
3 subjects_maths & engin not progr; NI	Logic\Sets\Counting	24	0.500	0.593
3 subjects, none_all 3; NI	Logic\Sets\Counting	52	0.461	0.647
Input set elements explicitly xeZ x lt b; WI	Logic\Sets\Elements	70	0.457	0.659
union__Input set elements explicitly xeN x le gt b; WI	Logic\Sets\Union	107	0.430	0.608
find (AnB) complement when (AnB) = { }; WI	Logic\Sets\Complement	75	0.427	0.628
3 subjects_ progr & engin not maths; NI	Logic\Sets\Counting	45	0.422	0.599
3 subjects_engin ; NI	Logic\Sets\Counting	54	0.389	0.599
3 subjects_maths & progr not engin; NI	Logic\Sets\Counting	44	0.364	0.705
3 subjects, none_maths & progr not engin; NI	Logic\Sets\Counting	43	0.349	0.299
union__Input set elements explicitly xeN x ge le b; WI	Logic\Sets	38	0.342	0.641
Input set elements explicitly xeZ abs(x) lt b; WI	Logic\Sets\Elements	47	0.340	0.590
given universal set, input set elements explicitly a lt x lt b; WI	Logic\Sets\Elements	24	0.333	0.492
3 subjects, none_2 subjects; NI	Logic\Sets\Counting	46	0.326	0.526
3 subjects_maths; NI	Logic\Sets\Counting	44	0.318	0.510
laws of set algebra; 4TFUSP	Logic\Sets\Algebra	322	0.317	0.491
3 subjects, none_maths; NI	Logic\Sets\Counting	45	0.311	0.535
Input set elements explicitly xeZ x le b; WI	Logic\Sets\Elements	71	0.310	0.619
3 subjects, none_progr; NI	Logic\Sets\Counting	37	0.297	0.633
3 subjects_progr; NI	Logic\Sets\Counting	32	0.250	0.374

2 operations; 4TFUSP	Logic\Sets\Algebra	186	0.242	0.545
3 subjects, none_ progr & engin not maths; NI	Logic\Sets\Counting	46	0.239	0.263
3 subjects_2 subjects; NI	Logic\Sets\Counting	42	0.238	0.470
3 subjects, none_engin ; NI	Logic\Sets\Counting	41	0.219	0.448
3 operations; 4TFUSP	Logic\Sets\Algebra	255	0.205	0.411
3 subjects, none_maths & engin not progr; NI	Logic\Sets\Counting	30	0.200	0.294
Input set elements explicitly xeZ abs(x) ge b; WI	Logic\Sets\Elements	40	0.175	0.523

The data on sets recommends that teachers should spend more time on the counting principle as well as revision of definitions of integers and absolute value.

9.2.3. Logic

In general students found questions on connectives (Figure 5.4) as well as on translation of an English statement to symbolic logic (Figure 5.2.2c) and vice-versa (Figure 5.2.2b) rather easy. However, these questions are of the multi-choice type, which heavily scaffold the students' thinking by suggesting the form of the answer, encouraging rechecking if they fail to get a displayed answer and rewarding partial knowledge or skills (although 'None of these' can be the correct response). This makes the questions much easier, which can be viewed positively, especially at the start of a module when one wants to build students' confidence. On the other hand, the multi-response questions on applications (Figure 5.3) have been correctly answered on average by 30-50% of participants, depending on the year. The difficulty here may be caused by the fact that the number of skills required is broad, or by the questions type. Despite the possible responses being provided, the number of required selections is not given. The questions on truth tables can be found at varying difficulty. Those containing operations such as NAND, NOR, XOR, and the backward arrow for the conditional statement (converse) are found to be more difficult.

Table 9.2.3: Combined questions from five years on logic, organised by their facility/mean score.

Question description	Topic	Times answered	Mean score	Correlation
ma*nb=0; m,n +ve MC	Logic\Translations\Connectives	583	0.849	0.423
p?(q?r): symbolic to words using implies; MC	Logic\Translations	156	0.827	0.351
(p?q)?r: words to symbolic using implies; MC	Logic\Translations	143	0.825	0.406
(NOTa)AND[[(b)OR(a)]AND (NOTb)] TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	20	0.800	0.850
(NOTa)OR[(NOTb)AND(b)] TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	23	0.782	0.836
[(a)OR(NOTb)]OR[(NOTb) OR(a)] TF input; WI	Logic\Truth Tables\2-3 operators of the same type, with parenthesis	22	0.773	0.827
p?q: words to symbolic using implies; MC	Logic\Translations	170	0.759	0.460
(NOTa)IFF[(b)OR[[(NOTb)IF(a)]AND(a)]] TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	8	0.750	0.983
p?q: symbolic to words using implies; MC	Logic\Translations	160	0.750	0.518
(a)IF[(NOTb)IF[(a)IF(b)]] TF input; WI	Logic\Truth Tables\2-3 operators of the same type, with parenthesis	24	0.750	0.673
(p?q)?r: symbolic to words using if; MC	Logic\Translations	157	0.732	0.420
[(NOTa)IFF(b)]IFF[(a)IFF(NOTb)] TF input; WI	Logic\Truth Tables\2-3 operators of the same type, with parenthesis	22	0.727	0.895
[(NOTa)AND[(b)OR(a)]]AND(NOTb) TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	18	0.722	0.748
[(a)OR(b)]OR(NOTb) TF input; WI	Logic\Truth Tables\2-3 operators of the same type, with parenthesis	28	0.714	0.783
[(a)OR(NOTb)]IFF(NOTc) TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	20	0.700	0.773
NOT[(a)OR(b)]OR[NOT[(a) AND(NOTc)]] TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	22	0.682	0.859
p?(q?r): words to symbolic using if; MC	Logic\Translations	172	0.680	0.474
[(a)IF(notb)]IF(b) TF input; WI	Logic\Truth Tables\2-3 operators of the same type, with parenthesis	27	0.666	0.732
(a)AND[(NOTb)AND(a)] TF input; WI	Logic\Truth Tables\2-3 operators of the same type, with parenthesis	21	0.666	0.777
(NOTb)OR[(NOTa)OR(b)] TF input; WI	Logic\Truth Tables\2-3 operators of the same type, with parenthesis	28	0.643	0.673

NOT[(a)IFF(b)]OR[(NOTc)AND(NOTb)] TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	14	0.643	0.864
[[(NOTb)AND(c)]AND(c)]AND(NOTa) TF input; WI	Logic\Truth Tables\2-3 operators of the same type, with parenthesis	25	0.640	0.698
(a)IF[(b)IF(c)] TF input; WI	Logic\Truth Tables\2-3 operators of the same type, with parenthesis	33	0.637	0.683
OR(NOT(a), OR(b, (NOT(AND(a, NOT(c))))) TF input; WI	Logic\Truth Tables	19	0.632	0.768
[(a)AND(NOTb)]OR[[(b)IF(NOTa)]IFF(NOTa)] TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	19	0.632	0.654
p?q: words to symbolic using if; MC	Logic\Translations	187	0.621	0.439
[(a)OR(NOTb)]IFF[NOT[(NOTa)AND(b)]] TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	26	0.615	0.718
(a)IF[(NOTb)IFF(NOTa)] TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	26	0.615	0.676
AND(AND(NOT(a), OR(b, a)), NOT(b)) TF input; WI	Logic\Truth Tables	20	0.600	0.454
NOT[(a)backIF(b)]AND[(NOTb)OR(NOTa)] TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	10	0.600	0.650
(a)IF[(NOTb)AND(NOTa)] TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	12	0.583	0.663
ma+nb=0; m,n +ve MC	Logic\Translations\Connectives	625	0.579	0.531
[(a)AND(NOTb)]IFF(b) TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	26	0.577	0.765
(NOTb)IF(NOTa)OR(a) TF input; WI	Logic\Truth Tables\2 operators of different type, without parenthesis	44	0.568	0.793
p?q: symbolic to words using if; MC	Logic\Translations	159	0.566	0.509
AND(NOT(a), AND(OR(b,a), NOT(b))) TF input; WI	Logic\Truth Tables	16	0.563	0.816
[(a)OR(b)]AND(NOTa) TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	20	0.550	0.638
(a)IF[[(b)IFF(NOTa)]backIF(NOTb)] TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	15	0.533	0.917
OR(NOT(a), AND(NOT(b),(b))) TF input; WI	Logic\Truth Tables	32	0.531	0.763
(a)IF(NOTb)AND(c) TF input; WI	Logic\Truth Tables\2 operators of different type, without parenthesis	55	0.527	0.684
NOT[(a)IF(b)]IF[(NOTb)IFF(NOTa)] TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	19	0.526	0.536

[(a)AND(notb)]AND(a) TF input; WI	Logic\Truth Tables\2-3 operators of the same type, with parenthesis	21	0.524	0.716
IFF(IFF(NOT(a), b),a) TF input; WI	Logic\Truth Tables	37	0.514	0.698
IF(a,IF(NOT(b), IFF(b,NOT(b))))) TF input; WI	Logic\Truth Tables	18	0.500	0.778
[(NOTc)backIF(b)]AND[(NOTa)IF(c)] TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	16	0.500	0.714
(a)IF{[(NOTb)OR[(a)IFF(b)]] backIF(NOTa)} TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	28	0.500	0.688
(a)AND(b)IFF(NOTc) TF input; WI	Logic\Truth Tables\2 operators of different type, without parenthesis	41	0.488	0.716
(b)IFF[(NOTa)IFF(NOTb)] TF input; WI	Logic\Truth Tables\2-3 operators of the same type, with parenthesis	35	0.486	0.679
IF(a, IFF(NOT(b),NOT(a))) TF input; WI	Logic\Truth Tables	25	0.480	0.766
IF(a, AND(NOT(b),NOT(a))) TF input; WI	Logic\Truth Tables	50	0.480	0.655
(NOTa)IFF(b)AND(NOTa) TF input; WI	Logic\Truth Tables\2 operators of different type, without parenthesis	50	0.480	0.597
OR(OR(a, b),NOT(b)) TF input; WI	Logic\Truth Tables	46	0.479	0.598
[NOT[(a)IFF(b)]IF(c)]IFF(NOTb) TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	23	0.478	0.762
(a)OR(b)IFF(NOTc) TF input; WI	Logic\Truth Tables\2 operators of different type, without parenthesis	42	0.476	0.711
(NOTa)IFF[(b)IF[(c)AND(NOTb)]] TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	21	0.476	0.773
(NOTa)OR{(b)OR[NOT[(a)AND(NOTc)]]} TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	19	0.474	0.845
NOT[(a)IF(b)]IFF(NOTa)TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	19	0.474	0.323
(NOTa)OR(NOTa)IF(NOTb) TF input; WI	Logic\Truth Tables\2 operators of different type, without parenthesis	36	0.472	0.853
Card turning; MR2/4	Logic\Translations\Arguments	559	0.460	0.560
[(NOTa)IFF(b)]IFF(a) TF input; WI	Logic\Truth Tables\2-3 operators of the same type, with parenthesis	33	0.454	0.763
IFF(OR(a, NOT(b)),NOT(c)) TF input; WI	Logic\Truth Tables	40	0.450	0.732
NOT[(NOTa)IFF(c)]NAND[(NOTb)AND(c)] TF input; WI	Logic\Truth Tables\3-5 operators, inc XOR,NAND,NOR; with parenthesis	38	0.448	0.537
[(NOTa)IFF(b)]AND[[(b)XOR(NOTa)]IF(b)] TF input; WI	Logic\Truth Tables\3-5 operators, inc	36	0.445	0.677

	XOR,NAND,NOR, with parenthesis			
NOT[(NOTa)NAND(b)]IF[(NOTb)OR[(a)IFF(b)]] TF input; WI	Logic\Truth Tables\3-5 operators, inc XOR,NAND,NOR, with parenthesis	41	0.439	0.682
NOT{(a)IFF[(NOTb)NOR[(b)AND(NOTa)]]}NAND[(a)backIF(b)] TF input; WI	Logic\Truth Tables\3-5 operators, inc XOR,NAND,NOR, with parenthesis	32	0.437	0.587
IFF(AND(a, NOT(b)),b) TF input; WI	Logic\Truth Tables	78	0.436	0.659
(NOTa)AND(NOTb)IF(a) TF input; WI	Logic\Truth Tables\2 operators of different type, without parenthesis	46	0.435	0.795
[(NOTb)NOR[(b)OR(NOTa)]]IF(a) TF input; WI	Logic\Truth Tables\3-5 operators, inc XOR,NAND,NOR; with parenthesis	42	0.429	0.630
[(a)IF[(b)IFF(NOTa)]]backIF(NOTb) TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	33	0.424	0.635
[[(a)AND(NOTb)]NAND(NOTa)]IF[(b)NOR(NOTa)] TF input; WI	Logic\Truth Tables\3-5 operators, inc XOR,NAND,NOR, with parenthesis	45	0.422	0.586
AND(a, AND(NOT(b),a)) TF input; WI	Logic\Truth Tables	45	0.422	0.792
Card turning; MR3/6	Logic\Translations\Arguments	564	0.421	0.498
NOT(IF(NOT(a), b)) TF input; WI	Logic\Truth Tables	88	0.409	0.685
NOT, AND, OR, IF, IFF, XOR, NAND, NOR TABLE TF input; WI	Logic\Truth Tables	86	0.407	0.732
AND(OR(a, b),NOT(a)) TF input; WI	Logic\Truth Tables	18	0.389	0.819
OR(NOT(OR(a, b)), NOT(AND(a, NOT(c)))) TF input; WI	Logic\Truth Tables	21	0.381	0.739
(a)IFF(b)OR(NOTa) TF input; WI	Logic\Truth Tables\2 operators of different type, without parenthesis	43	0.372	0.744
OR(OR(NOT(a),b),(NOT(AND(a, NOT(c))))) TF input; WI	Logic\Truth Tables	27	0.370	0.674
OR(a,AND(NOT(b),c)) TF input; WI	Logic\Truth Tables	22	0.364	0.773
(c)AND[[(NOTa)NOR(b)]OR(NOTc)] TF input; WI	Logic\Truth Tables\3-5 operators, inc XOR,NAND,NOR; with parenthesis	47	0.361	0.693
IFF(NOT(a),(OR(b),(AND(IF(NOT(b),a),a))) TF input; WI	Logic\Truth Tables	39	0.359	0.761
IFF(IF(NOT(a), b),NOT(c)) TF input; WI	Logic\Truth Tables	53	0.359	0.529
IF(a, IF(b,c)) TF input; WI	Logic\Truth Tables	35	0.343	0.570

IF(NOT(IF(a, b)),IFF(NOT(b),NOT(a))) TF input; WI	Logic\Truth Tables	44	0.341	0.642
AND(OR(a, NOT(b)),c) TF input; WI	Logic\Truth Tables	15	0.333	0.508
AND(AND(NOT(a), OR(a, b)), NOT(b)) TF input; WI	Logic\Truth Tables	61	0.311	0.586
[(b)backIF(NOTa)]AND[(a) OR[(b)IF(a)]] TF input; WI	Logic\Truth Tables\2-4 operators of different type, with parenthesis	13	0.307	0.755
OR(NOT(IFF(a, b)),AND(NOT(c),NOT(b))) TF input; WI	Logic\Truth Tables	42	0.286	0.644
IF(NOT(b),(IF(a, IFF(b,NOT(b))))) TF input; WI	Logic\Truth Tables	87	0.276	0.651
NOR(NOT(b), OR((b),NOT(a))) TF input; WI	Logic\Truth Tables	89	0.269	0.723
OR(NOT(OR(a, b)),AND(NOT(b),NOT(a))) TF input; WI	Logic\Truth Tables	31	0.258	0.738
OR(NOT(AND(a, b)),AND(c, NOT(b))) TF input; WI	Logic\Truth Tables	39	0.257	0.734
AND(IF(b, NOT(c)), IF(NOT(a), c)) TF input; WI	Logic\Truth Tables	83	0.241	0.516
NAND(NOT(IFF(NOT(a), c)), NOT(b)) TF input; WI	Logic\Truth Tables	77	0.234	0.565
NAND(NOT(IFF(a), NOR(NOT(b), AND(b, NOT(a)))), IFF(b,a)) TF input; WI	Logic\Truth Tables	83	0.229	0.547
IFF((IF(NOT(IFF(a, b)),c)),NOT(b)) TF input; WI	Logic\Truth Tables	42	0.214	0.384
OR(AND(NOT(d), NOR(NOT(a), b)), NOT(c)) TF input; WI	Logic\Truth Tables	68	0.206	0.618
IFF(NOT(a),IF(b,(AND(c, NOT(b)))) TF input; WI	Logic\Truth Tables	31	0.161	0.573
IFF(NOT(IF(a, b)),NOT(a)) TF input; WI	Logic\Truth Tables	37	0.135	0.724

Results in the table above suggest that questions on applications and truth tables involving NAND, XOR, NOR and backward arrow for implication should be highlighted more to students.

9.2.4. Linear programming

As described in Chapter 6, questions on linear programming are of the multi-choice or (various) numerical input types. Although questions are testing different skills, it is clear that students were more successful when answering MC questions on feasible regions (Figure 6.1). No significant differences can be seen from the question style. However, students were doing marginally better when they were asked to find inequalities for a given scenario, rather than graphs. This means that their preference was to analyse the English text, rather than the graph for which they had to work out the equations of straight lines when given graphically. Questions on finding the optimal solution (see Figure 6.2) were found to be harder, having a lower mean score. The difficultly here seems to relate to applying sequential tasks correctly, in this case formulating the feasible region and objective function and understanding the roles of their slopes in determining and evaluating the optimal point, or having problems with the evaluation of the objective function at each vertex of the feasible region.

Table 9.2.4: Combined questions from five years on linear programming, organised by their facility/mean score.

Question description	Topic	Times answered	Mean score	Correlation
given scenario find matching inequalities, wrong coeffs, region IV; MC	Decision mathematics\ Linear programming\ Feasible regions	256	0.781	0.501
given scenario find matching inequalities, region IV; MC	Decision mathematics\ Linear programming\ Feasible regions	284	0.757	0.577
given graph find matching inequalities, region II; MC	Decision mathematics\ Linear programming\ Feasible regions	245	0.710	0.632
given graph find matching inequalities, region II,wrong coeffs; MC	Decision mathematics\ Linear programming\ Feasible regions	264	0.693	0.647
given graph find matching inequalities, region III; MC	Decision mathematics\ Linear programming\ Feasible regions	280	0.682	0.657
given graph find matching inequalities, region IV,wrong coeffs; MC	Decision mathematics\ Linear programming\ Feasible regions	272	0.680	0.643
given graph find matching inequalities, region IV; MC	Decision mathematics\ Linear programming\ Feasible regions	270	0.674	0.659
given graph find matching inequalities, region III,wrong coeffs; MC	Decision mathematics\ Linear programming\ Feasible regions	233	0.639	0.712
given graph find matching inequalities, region I; MC	Decision mathematics\ Linear programming\ Feasible regions	265	0.615	0.704
given graph find matching inequalities, region I,wrong coeffs; MC	Decision mathematics\ Linear programming\ Feasible regions	269	0.602	0.659
given inequalities find graph region; MC	Decision mathematics\ Linear programming\ Feasible regions	267	0.596	0.661
given scenario find optimum X (possibly incl degeneracy), region IV; RNI	Decision mathematics\Linear programming\Optimisati on (any method)	467	0.375	0.713

given scenario find optimum X (no degeneracy), region IV; RNI	Decision mathematics\Linear programming\Optimisation (any method)	568	0.304	0.710
given scenario find optimum X position (no degeneracy), region IV; 2NI	Decision mathematics\Linear programming\Optimisation (any method)	503	0.283	0.669

The table clearly indicates that students should have more time dedicated to questions on optimisation. Perhaps one should ask students to submit their workings on paper in order to identify which of the suggested above potential problems is the issue.

9.2.5. Graph theory

Questions on graph theory that students find relatively easy include those which ask them to input the vertex set (Figure 7.3), edge set, degree sequence (Figure 7.2), with no clear trend on the question style showing either the graph or its matrix representation. Students also did well on finding the graph representation of an adjacency matrix or vice versa (Figure 7.1.3), or on finding simple and connected graph. The questions of lower mean score include those on (minimum) spanning trees (Figure 7.4, Figure 7.4.3). Whether they are of MC or WI type, using adjacency matrix or graphs, finding a spanning tree or a minimum spanning tree, was found to be difficult.

Table 9.2.5: Combined questions from five years on graph theory, organised by their facility/mean score.

Question description	Topic	Times answered	Mean score	Correlation
Given connected graph_input vertex set; WI+check	Decision mathematics\ Graph theory\Edge and vertex sets\Vertex sets	153	0.843	0.467
degree sequence of the graph (simple, connected graph); WI	Decision mathematics\ Graph theory\Degree\ Degree sequence	29	0.759	0.667
degree sequence of the adjacency matrix (with multi edges and loops); WI	Decision mathematics\ Graph theory\Degree\ Degree sequence	28	0.750	0.228
Given graph, input vertices; WI+check	Decision mathematics\ Graph theory\Edge and vertex sets	181	0.729	0.511
Given adjacency matrix, find matching graph; MC	Decision mathematics\ Graph theory\Adjacency matrices	191	0.728	0.455
Find the simple connected graph given the graphs; RandMC	Simple and connected graphs	318	0.723	0.399
degree sequence of the graph	Decision mathematics\	36	0.694	0.572

(simple, disconnected graph); WI	Graph theory\Degree\ Degree sequence			
degree sequence of the graph (with multi edges); WI	Decision mathematics\ Graph theory\Degree\ Degree sequence	32	0.688	0.583
What is wrong with the adjacency matrix; RWI+check	Decision mathematics\ Graph theory\Adjacency matrices	421	0.686	0.638
Sum of entries (Introduction to Degree); NI	Decision mathematics\ Graph theory\Degree	172	0.669	0.472
degree sequence of the adjacency matrix (simple, connected graph); WI	Decision mathematics\ Graph theory\Degree\ Degree sequence	30	0.667	0.790
degree sequence of the adjacency matrix (simple, disconnected graph); WI	Decision mathematics\ Graph theory\Degree\ Degree sequence	37	0.649	0.559
Find the simple connected graph given the graphs or adjacency matrices; RandMC	Simple and connected graphs	318	0.648	0.482
outdegree of the vertex of the adjacency matrix; NI	Decision mathematics\ Graph theory\Degree\ In&out degree	34	0.647	0.373
Given disconnected graph_input vertex set; WI+check	Decision mathematics\ Graph theory\Edge and vertex sets\Vertex sets	115	0.643	0.738
Given graph, find matching adjacency matrix; MC	Decision mathematics\ Graph theory\Adjacency matrices	422	0.640	0.524
Given graph with loops_input edge set; WI+check	Decision mathematics\ Graph theory\Edge and vertex sets\Edge sets	125	0.640	0.694
Given graph, input vertices (with disconnected vertices); WI+check	Decision mathematics\ Graph theory\Edge and vertex sets	173	0.630	0.557
degree sequence of the graph (with multi edges and loops); WI	Decision mathematics\ Graph theory\Degree\ Degree sequence	29	0.621	0.306
Find the simple connected graph given the adjacency matrices; RandMC	Simple and connected graphs	357	0.608	0.489
Given simple, connected graph_input edge set; WI+check	Decision mathematics\ Graph theory\Edge and vertex sets\Edge sets	109	0.606	0.673
degree sequence of the adjacency matrix (with multi edges); WI	Decision mathematics\ Graph theory\Degree\ Degree sequence	33	0.576	0.528
Given digraph, input edges; RWI+check	Decision mathematics\ Graph theory\Edge and vertex sets	150	0.567	0.652
which is the n'th edge of the minimum spanning tree_5-6 vertices_Prim's algorithm; WI	Decision mathematics\ Graph theory\Spanning trees\Minimal spanning trees\Prims	68	0.544	0.570
which is the n'th edge of the minimum spanning tree_5-6 vertices_Kruskal's algorithm; WI	Decision mathematics\ Graph theory\Spanning trees\Minimal spanning trees\Kruskal	47	0.511	0.521
Determining degree; NI	Decision mathematics\Graph theory\Degree	145	0.490	0.520
which is the n'th edge of the	Decision mathematics\	57	0.474	0.617

minimum spanning tree_7 vertices_Kruskal's algorithm; WI	Graph theory\Spanning trees\Minimal spanning trees\Kruskal			
indegree of the vertex of the adjacency matrix; NI	Decision mathematics\ Graph theory\Degree\ In&out degree	34	0.471	0.592
Spanning trees using graphs for an adjacency matrix; RandMC	Decision mathematics\ Graph theory\Spanning trees	128	0.453	0.472
degree of the vertex of the network matrix (symmetric graph); NI	Decision mathematics\ Graph theory\Degree\ In&out degree	41	0.439	0.524
Shortest distance between two towns; RNI	Decision mathematics\ Graph theory\Shortest path	535	0.430	0.469
outdegree of the vertex of the network matrix of a digraph; NI	Decision mathematics\ Graph theory\Degree\ In&out degree	23	0.391	0.262
Generate the degree sequence; RWI	Decision mathematics\ Graph theory\Degree	156	0.385	0.614
which is the n'th edge of the minimum spanning tree_7 vertices_Prim's algorithm; WI	Decision mathematics\ Graph theory\Spanning trees\Minimal spanning trees\Prims	78	0.359	0.510
was AB edge added/ rejected/not considered and at what step_5-6 vertices_Kruskal's algorithm; WI	Decision mathematics\ Graph theory\Spanning trees\Minimal spanning trees\Kruskal	53	0.358	0.627
Number of vertices in a partite set of a bipartite graph; NI	Decision mathematics\ Graph theory\Bipartite graphs	154	0.351	0.658
Spanning trees using adjacency matrices; RandMC	Decision mathematics\ Graph theory\Spanning trees	146	0.336	0.478
Spanning trees using graphs; RandMC	Decision mathematics\ Graph theory\Spanning trees	151	0.325	0.531
Given graph, input edges; WI+check	Decision mathematics\ Graph theory\Edge and vertex sets	169	0.314	0.452
what is the weight of the minimum spanning tree; NI	Decision mathematics\ Graph theory\Spanning trees\Minimal spanning trees	61	0.295	0.544
what is the minimum spanning tree_5-6 vertices_Prim's algorithm; WI	Decision mathematics\ Graph theory\Spanning trees\Minimal spanning trees\Prims	94	0.287	0.564
Indegree and Outdegree; RNI	Decision mathematics\ Graph theory\Degree	143	0.280	0.592
was AB edge added/rejected/not considered and at what step_5-6 vertices_Prim's algorithm; WI	Decision mathematics\ Graph theory\Spanning trees\Minimal spanning trees\Prims	79	0.278	0.417
Bipartite graph / adjacency matrix search; MC	Decision mathematics\ Graph theory\Bipartite graphs	147	0.272	0.375
what is the minimum spanning tree_5-6 vertices_Kruskal's algorithm; WI	Decision mathematics\ Graph theory\Spanning trees\Minimal spanning	71	0.268	0.737

	trees\Kruskal			
Bipartite graph search; MC	Decision mathematics\ Graph theory\Bipartite graphs	128	0.258	0.415
Spanning trees using adjacency matrices for a graph; RandMC	Decision mathematics\ Graph theory\Spanning trees	144	0.250	0.322
was AB edge added/rejected/not considered and at what step_7 vertices_Kruskal's algorithm; WI	Decision mathematics\ Graph theory\Spanning trees\Minimal spanning trees\Kruskal	48	0.250	0.469
minimum spanning tree; RWI	Decision mathematics\ Graph theory\Spanning trees\Minimal spanning trees	55	0.236	0.546
Bipartite adjacency matrix search; MC	Decision mathematics\ Graph theory\Bipartite graphs	136	0.235	0.417
what is the minimum spanning tree_7 vertices_Kruskal's algorithm; WI	Decision mathematics\ Graph theory\Spanning trees\Minimal spanning trees\Kruskal	49	0.224	0.630
was AB edge added/rejected/not considered and at what step_7 vertices_Prim's algorithm; WI	Decision mathematics\ Graph theory\Spanning trees\Minimal spanning trees\Prims	79	0.215	0.483
what is the minimum spanning tree_7 vertices_Prim's algorithm; WI	Decision mathematics\ Graph theory\Spanning trees\Minimal spanning trees\Prims	76	0.211	0.536
indegree of the vertex of the network matrix of a digraph; NI	Decision mathematics\ Graph theory\Degree\ In&out degree	30	0.200	0.577
what is the n'th edge of the minimum spanning tree_using Prim's algorithm; WI	Decision mathematics\ Graph theory\Spanning trees\Minimal spanning trees	66	0.197	0.475

Although the number of questions attempted is smaller than the majority of the others than have been available for many years, comparison of questions that test Prim's or Kruskal's algorithms with the overall picture shown in Figure 9.4.1 show generally lower facility [0.197,0.544] and consequently below average discrimination for most cases. This indicates that the questions are more challenging for students, requiring them to be able to implement the algorithm rather than using some other method: that is to say, the questions appear to test the required skills accurately and hence represent a useful contribution to the database and indeed to the whole methodology of designing and coding such type of questions that is more widely applicable.

As stated above and shown in Table 9.2.5, more time should be spent on (minimum) spanning trees and their algorithms. The table also suggests that students have problems with understanding the weights in a network matrix; it should be stressed that these numbers are not the number of edges, as in an adjacency matrix, but the weight of a single edge.

9.3. Discrimination

Discrimination is a correlation; it is "a number that ranges from -1.0 to +1.0 and illustrates how the results on this question compare to results on the test overall' (Shepherd and Godwin, 2004). In the CAA analysis this piece of information is called correlation. By looking at the data for each question it is possible to tell how well it would help distinguish participants who did well on the test and those who did badly: "*Positive value = those who got the item correct also did well on the exam, and those who got the item wrong also did poorly on the exam. Negative value = those who did well on the test got the item wrong, those who did poorly on the test got the item right. +0.10 or above is typically required to keep on item.*" (Peterson, 2013, where '*item*' refers to a question here). Unclear questions or incorrectly coded correct answers or distractors are possible reasons for negative values and would require fixing. Values around zero are when there is no correlation between the marks scored for the question and in the test overall. The higher the positive values are the stronger the correlation is.

To analyse questions according to their discrimination the same tables as in the previous section were used but ordered according to the correlation (last column); the spreadsheet allowing the reader to do this is included in Appendix C. For each topic it includes combined years and for each year. For this section, please refer to the tables above or in Appendix C for values.

The majority of questions of all types and topics have a discrimination greater than 0.5 which implies they are well designed. Before combining the same questions from different years some of them had a negative correlation or close to zero. However, they were single cases, specific to single years and therefore combining years allowed anomalies to be eliminated but still resulted in some questions having low positive correlation values. These were examined for errors or ambiguities but none we found. So the reason is unclear – it may be that students were simply guessing, but that begs the question of why.

9.3.1. Numbers

In all years, the maximum value of correlation is around 0.8, however after combining questions from five years the highest value decreased to 0.716, and it is on addition of octal numbers when an addition table was available. The lowest correlation of all years occurred in academic years finishing in May 2009 and 2010 and had a value of around 0.1 but this increased to 0.3 in 2013. In 2012-13 questions on significant figures were introduced and one question resulted in a discrimination of 0.005 having the lowest entry in the dataset. Throughout the years, the most discriminating questions involved the conversion of a base 3 to 9 number into a decimal representation or vice versa. Highly discriminating questions are also those on modular arithmetic (Figure 3.2). The data varied during the course of five years but overall the least discriminating are those on conversions: ‘binary to decimal’ or ‘decimal to scientific notation’ and vice versa with a high mean score saying that both strong and weak students got the question right. On the other hand, questions asking for subtraction using either ‘ones complement’ or some questions on significant figures are of low discrimination since even good students got the questions wrong. Questions on non-decimal arithmetic are distributed throughout the rank ordering but usually take high correlation values.

9.3.2. Sets

Some questions on sets have very high correlation, exceeding 0.9 in some years and therefore resulting in values greater than 0.8 after combining information from all five years. On the other hand, nine questions, in some years had a negative or zero correlation but after merging the data on them from other years the values increased and the lowest correlation had a value of 0.263. Therefore, overall, questions are well discriminating. There is no consistency between the questions on the counting principle (Figure 4.12). In some years discrimination is low while in others these questions were highly discriminating. This results in most of them having lower correlation than other questions on sets. As mentioned in Section 9.2.2, these questions had low mean score. This means that due to the high difficulty, most students get the questions wrong. Questions on binary arithmetic of sets or listing the set elements explicitly when given in a set-builder notation have varying correlation.

9.3.3. Logic

Questions on logic are more discriminating than those on numbers and sets. Only three questions in single years had a non-positive correlation. Either this applied just for a particular cohort or the questions were corrected in one of many improvement processes. This led to the minimum correlation of 0.403 in the 2012-13 academic year and 0.323 in the combined dataset. The least discriminating questions are those on translation from symbolic logic into English statements and vice versa (Figures 5.2.2), with no specific difference with respect to the number of operators involved. Questions on arguments and connectives have a relatively low correlation in comparison to some of the questions on truth tables. Overall, the most discriminating questions are on truth tables with 2-4 operators of different type and with parenthesis (Figures 5.1.2).

9.3.4. Linear programming

All questions on linear programming are highly discriminating. After combining the data from all available years the correlation varies between 0.501 and 0.713. Only one question had a correlation slightly below 0.5 in three academic years. It is a multiple choice question on finding the inequalities matching to a given scenario when some of the coefficients are wrong (Figure 6.1.3). After merging all the questions for the five years, it is clearly seen that questions on optimisation (Figure 6.2) are more discriminating.

9.3.5. Graph theory

Questions on graph theory are correctly constructed since all of them have a positive correlation when looking at the data for each year and also after merging them. Questions on degree tend to have low correlation. Exceptions are two questions that ask for the degree sequence of the graph or the adjacency matrix of a simple, connected graph. Questions on edge and vertex sets of disconnected graphs, digraphs (Figure 7.3.3) or graphs with loops (Figure 7.3) are relatively more discriminating than those involving simple connected graphs. Questions on (minimum) spanning trees are generally in the middle range with the highest correlation being 0.737.

9.4. Facility vs. Discrimination

In the previous two sections, questions were analysed with respect to their facility and discrimination by looking at each of the five topics after the five years' worth of data was merged but also including some information for separate years. Some results were the same for the merged set of years, whereas others were different. Merging the data allowed the smoothing of possible anomalies for certain cohorts of students and also provided a wider picture. This section will show the distribution of questions by comparing them according to their type or topic. For an easier visualisation, scatter graphs will be shown.

9.4.1. By question type

This section will discuss whether the questions' facility or discrimination depends on the question type. Combined data was used to generate a scatter graph (Figure 9.4.1) in order to examine it. The diagram contains all question types together; however, graphs for each question type separately are available in Appendix C.

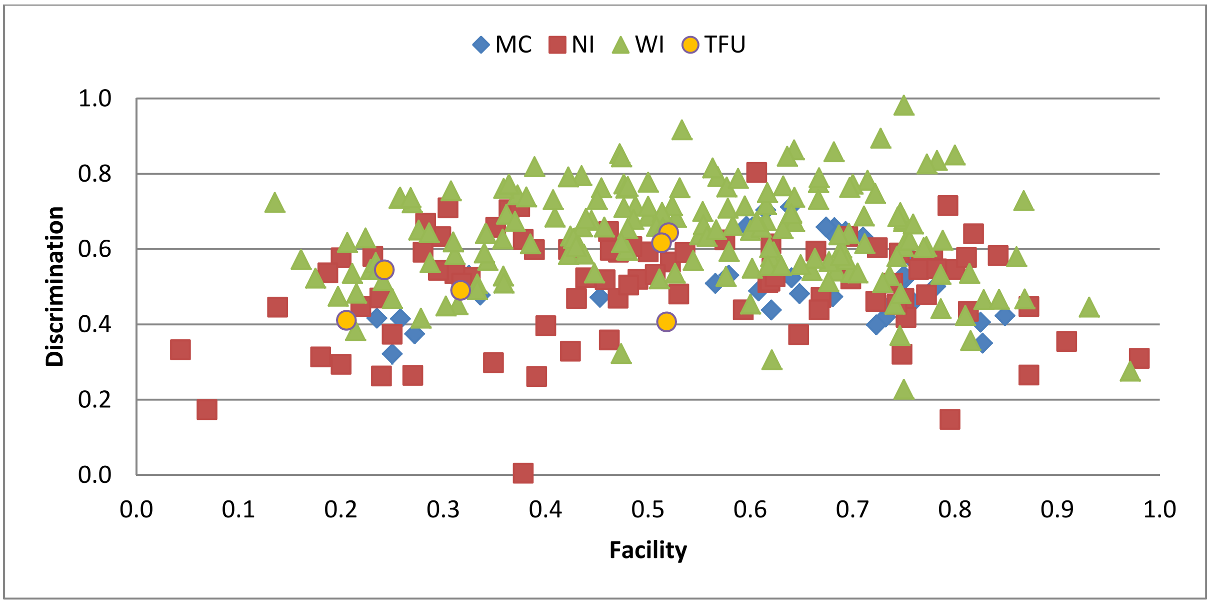


Figure 9.4.1: Scatter graph comparing the facility and discrimination of different question types. Blue squares represent multiple choice questions; red squares represent numerical input, responsive numerical input, as well as multiple numerical input type questions; green triangles represent word input or responsive word input type questions; whereas, orange circles represent true/false/undecidable type questions.

There is no substantial difference between questions of MC and NI types, but there is a slight increase in discrimination for WI-type questions. (There are too few TFU questions to draw any conclusions.) This means that calling questions at random in a test is reasonable and one does not have to worry too much about selecting by question type. Of course this also depends on the purpose of the test itself, especially if the assessment regime is norm or criterion referenced. The few anomalies (questions with either of the two parameters much different from those of remaining data) are for the questions that were run in a single year, with the exception of a WI type question with the highest discrimination that was run in the last two years. Moreover, these NI and WI type questions were answered only a few times. For the other questions, more data was available and hence the data was more robust. This means that the questions are constructed correctly and the correlation is more valid when the bigger sample size is taken into account. All the questions with results much different from the other questions were checked again and no mistakes have been found in the question design or coding. Furthermore, the majority of available questions have a correlation greater than 0.5 indicating that they discriminate well, they are well designed, and therefore the questions can be considered to be very robust and useful.

The most populous question type is Word Input comprising 184 questions. These questions strongly are located in the top part of the scatter graph indicating that they discriminate most. 87% of the WI type questions have a correlation greater than 0.5. 58% of these questions were answered correctly. The facility takes values between 0.135 and 0.971 while the discrimination is between 0.228 and 0.983.

The second most populous type of question is NI comprising 88 questions. Compared to questions of other types these are least discriminating on average. However, taking into account how many of them (60%) have correlation greater than 0.5 one could say they are the second most discriminating group. Both facility and discrimination have the most extreme values and vary between [0.043,0.98] and [0.005,0.804] respectively.

MC questions, comprising 36 questions, are divided into two groups according to their facility. Those that were answered correctly less than half the time are on bipartite graphs or spanning trees, whereas the other MC questions have facility above 0.5. The smallest mean score for a question was 0.235 and the highest 0.849. The range of discrimination is much lower (0.39) with the lowest value being 0.322 and the highest 0.712.

There were only six TFU questions so no general conclusions can be drawn. They are relatively hard as the highest mean score was 0.518 (lowest = 0.205). The reason for the TFU questions being consistently harder is probably due to the marking scheme requiring all answers to be correct for the mark. This is to offset the fact that the choice is limited so random guessing will result in getting some of the four inputs correct. The discrimination varies between 0.407 and 0.644.

9.4.2. By topic

To discuss whether the questions' facility or discrimination depends on the topic, another scatter graph was produced (Figure 9.4.2). This will allow the display of any trends; we see that there are no significant differences between the performance of questions of different topics.

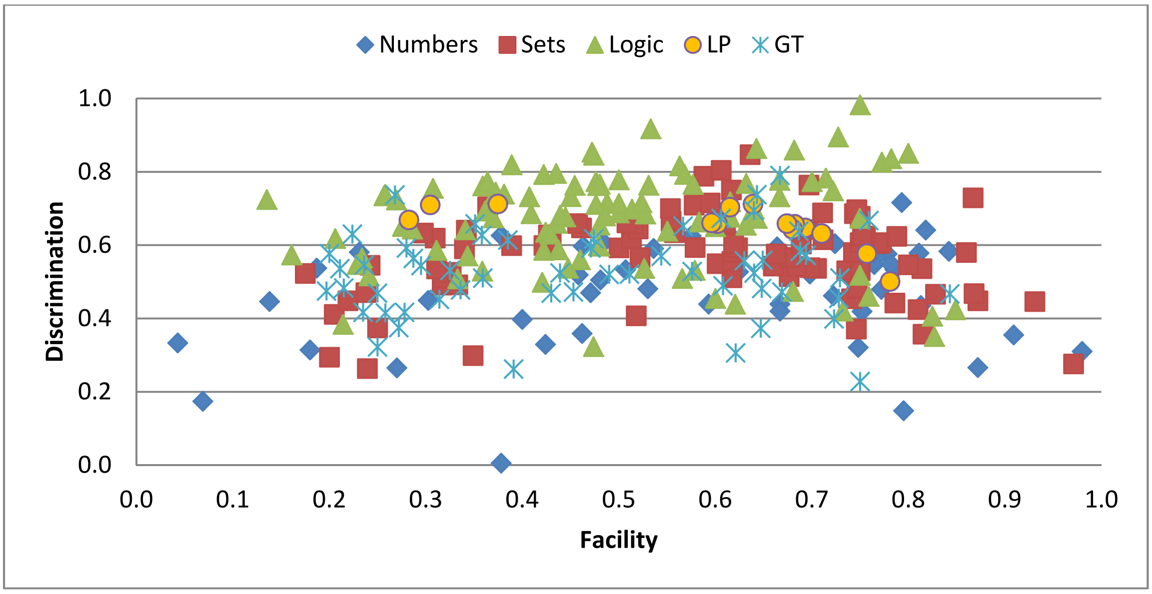


Figure 9.4.2: Scatter graph comparing the facility and discrimination of question for each topic. Blue squares represent questions on numbers, red squares – sets, green triangles – logic, orange circles – linear programming, blue stars – graph theory.

The questions separated from the rest are mainly on numbers, with one on sets that has high facility and other highly discriminating on logic. The possible reasons for this situation were described in the previous section.

Most of the questions on numbers have discrimination between 0.4 and 0.6, with a few just above 0.6. This makes questions from this topic the least discriminating of all. Discrimination takes values between 0.005 and 0.716, while facility between 0.043 and 0.98. 67% of the questions have a facility greater than 0.5.

Questions on sets are more discriminating. Most of them are between 0.5 and 0.7, with 76% above 0.5. Questions difficulty is comparable to those of numbers with 70% of questions answered correctly more than half the time. Overall, the values are not so spread out compared to those on numbers, as discrimination varies between [0.263,0.847] and facility between [0.175,0.971].

For logic the discrimination of the questions is high with the majority of the questions having values between 0.6 and 0.8 and 89% of the questions having the discrimination above 0.5. No questions are particularly separated from the group with discrimination between [0.323,0.983] and facility between [0.135,0.849]. Questions on logic are most difficult, with only 43% having a mean score above 0.5.

Although questions on linear programming do not have values of the discrimination greater than 0.8, most of them are between 0.6 and 0.8. This indicates that questions from this topic are very discriminating. However, there is the least number of questions available to students and more would be needed to draw a more general picture. Facility splits questions into two groups. Those having values below 0.4 are on optimisation whereas all but one of the questions of feasible regions have a facility above 0.6. Mean scores are between 0.283 and 0.781.

Graph theory is the second most difficult topic according to the data. Only 45% of the questions have a mean score greater than 0.5. Most of the questions with facility below 0.4 are on bipartite graphs and (minimum) spanning trees. Questions do not differ much with facility between 0.197 and 0.843, while discrimination is between 0.228 and 0.79, resulting in graph theory being the second least discriminating topic.

10. Analysis of exam scripts

Chapters 3-7 described the CAA questions students were solving as a part of their assessment. Chapter 8 presented students' perception on the CAA, while Chapter 9 used the extensive set of CAA answer files to evaluate students' engagement with the five topic areas and the discrimination and facility of the questions. These chapters lead up to the analysis of exam scripts and the possible impact of CAA on students' performance in their examinations.

Exam scripts were looked at several ways and for different reasons. The first was to determine common mistakes students make to:

- address teaching and teaching materials to meet students' needs;
- inform the assessment of specific skills;
- underpin the process of CAA design, as described below.

The exam questions were analysed with respect to indicators that appeared in most of the exam papers in years 2006-2012, and also in the CAA. Later, the exam results from seven years were compared for each topic (where data was available) with the mean number of questions answered on each topic in CAA, regarded as a measure of engagement with the topics. The chapter finishes with the assessment of the correlation between CAA and exam results.

This chapter is an extension of the study presented in 'Assessment of Elementary Discrete Mathematics' by Zaczek and Greenhow (2011).

10.1. Common errors

During my experience in teaching at secondary and higher education levels, I observed that identifying common mistakes students make is often possible. It is done by the analysis of homework, work done in class and in exams. Knowing what the possible students' misconceptions are may help to improve teaching since then the teacher can point them out to students as a precaution and/or spend more time on them. Teachers generally provide students with many examples to be solved so they can improve through practice and avoid mistakes. This takes place when the topic is introduced for the first time, but also as a form of preparation for the examination, assuming that after a period of time students may make them again.

It is easy to mark students' work as right or wrong. However, very often it is difficult or even sometimes impossible to understand students' thinking behind their given answers. Exam scripts of students taking Discrete Mathematics at Foundation Level in 2006 – 2012 were analysed and some common errors were found. They were categorised as: conceptual, notational, and terminological. The most frequently-occurring errors in each topic are presented in Table 10.1.

Table 10.1: Most common mistakes found in the exam scripts for each topic and categorised as conceptual, notational, terminological

Numbers		
Conceptual	*Notational*	*Terminological*
• Incorrectly writing remainders (typing what the calculator shows) • Adding octal numbers in a way you add decimals		
Sets		
Conceptual	*Notational*	*Terminological*
• Confusing union, intersection and difference • Thinking that if two non-empty sets are disjoint, then the difference between them is an empty set	• Not using brackets or using round brackets when listing the set's elements • Using wrong notation for an empty set e.g. {ø}	• Even numbers (not including negative numbers and 0) • Prime numbers (not including 2; including 1 and negative numbers) • Perfect squares (not including 0 and 1; including negative numbers)
Logic		
Conceptual	*Notational*	*Terminological*
• Incorrectly translating symbolic logic into English statements and vice versa • Rewriting truth values in truth tables with no negations where necessary	• Wrong number of rows in truth tables and therefore missing or repeating truth values • Confusion between symbols representing 'and' and 'or'	
Linear Programming		
Conceptual	*Notational*	*Terminological*
• Shading the wrong area on the graph for inequality	• Writing equalities instead of inequalities	
Graph Theory		
Conceptual	*Notational*	*Terminological*
• Weights in network matrix treated as number of edges • Confusion between Prim's and Kruskal's algorithms • Confusion between path and trail	• Drawing two loops when there is a '2' in adjacency matrix on diagonal	

These common mistakes can be included in the teaching by placing more emphasis on issues that are confusing students. This has the potential to increase the quality of teaching in the long run, especially if errors can be codified and shared between teaching staff. What might be needed is a commonly-accepted ‘taxonomy of errors’, perhaps along the lines of Dawkins (2006) or Schechter (n.d.), but this seems to be missing.

What is also good about such clearly-defined errors is that they can be programmed into the questions. Initially these were simply based on the structure of the questions themselves suggesting possible errors, and on my teaching experience and that of Dr Greenhow. This pragmatic approach was then augmented by the above evidence to improve the design and coding of multiple choice or responsive numerical/word input type questions. These question types require wrong, but realistic, answers and bespoke feedback matching the submitted answer. Given the validity of this design approach, more errors could and should be identified by looking at students’ CAA inputs. In cases when they are understandable (not when they were simple guesses) these could also be implemented in the CAA development cycle.

10.2. Indicators

Following Gill’s (2007) methodology, exam scripts have also been analysed using four (easily-identifiable) indicators. These are the ability to: write remainders; convert binary, octal, hexadecimal numbers; use curly brackets for sets; lay out a truth table with three statements so that TTTTFFFF, TTFFTTFF, TFTFTFTF are the columns (complete data in Appendix D). These indicators were selected because they are necessary, but not sufficient, for deeper student learning. Gill used similarly mundane identifiers in her study of A-level and first-year mechanics, namely the: usage of diagrams, logical presentation of solution, use of units, correct stating vectors. The findings are presented in Figure 10.2. Part of our purpose here was to find out if the CAA helps students to improve and seven years’ worth of data should help to examine this.

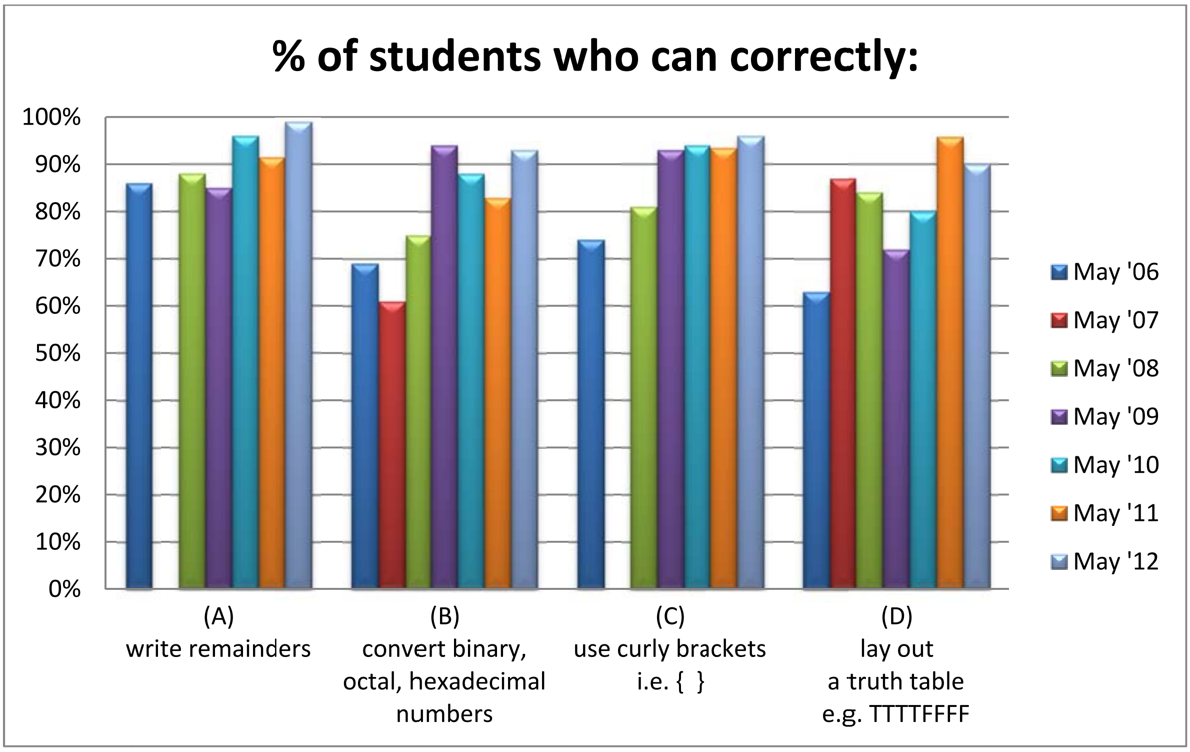


Figure 10.2: A year-by-year comparison of students' ability to: (A) write remainders; (B) convert binary, octal and hexadecimal numbers; (C) use curly brackets in sets; (D) lay out a truth table in a certain way. CAA was used prior to May 2009 exam.

During academic years finishing in 2006, 2007 and 2008 students did not have access to CAA and were taught by the same teacher. From 2008/09 to 2011/12 another staff member took over and used the CAA developed in Chapters 3-7. In addition, algebra and study skills disappeared, replaced by linear programming and graph theory. There were, however, no other significant changes to exam standards, entry requirements, time allocated to each topic or teaching style.

The exam scripts were also looked at in respect to some other indicators. However, no other ones were consistently identified throughout the years. This is mainly due to a change in the syllabus that limited possible indicators to be from the topics of numbers, sets or logic. Moreover, their choice had to be free from personal judgement and straightforward to determine whether students possess certain abilities or not. Due to these constraints, only four indicators were chosen. This is not to say that other indicators could not have been identified or chosen. Indeed, in the Recommendations, the further examination of exam scripts to identify other indicators is recommended. There are, nevertheless, two gaps in Figure 10.2, as it was difficult to identify two of the indicators on the exam in May 2007.

Overall, these indicators suggest a positive message about CAA. Indicators (B) and (C) show significant improvement when CAA was used.

- Before the software was used the lowest score for students being able to convert numbers to different base was 61% and the highest 75%, while after CAA being introduced the lowest was 83% and the highest 94%. This results in at least an 8% increase but reaches 33% when comparing the worst and the best results. The feedback screen for the questions on converting numbers between different bases includes a complete explanation that evidently had a positive impact on students.

- The number of students using curly brackets for sets also considerably increased. In 2006 74% of students were using curly brackets, while in 2008 it was 81%. These figures were higher in the last four years with values between 93% and 96%. Looking at the questions on finding for example the union or intersection of two sets, there are nine curly brackets (i.e. { }) which are also displayed on the feedback screen in addition to the solution that usually displays even more of them. In this way students are continuously exposed to the correct notation and are required to use it in submitted answers.

- A not so significant, but also a visible improvement in the students' performance was observed in writing remainders (indicator (A)). The bar chart shows that the results were always high (above 80%) but after a dip in 2009 there was increase to above 90%. In 2012 99% of students attempting the question on converting the decimal numbers into numbers of different base were able to write the remainders correctly.

- It is impossible to claim an increase in the number of students laying out the truth values for three statements in a way that columns are: TTTTFFFF, TTFFTTFF, TFTFTFTF. In order to avoid ambiguity, questions presented to students had the truth tables partially completed with this information. The bar chart suggests that CAA did not have an impact on students since the results varied both before and after the CAA was used.

Summarising, after CAA was used, three out of four selected indicators showed an increase in the number of students answering questions correctly/in a certain way. These three skills were also performed in the CAA directly. Students had to find remainders, convert numbers to different bases, and use curly brackets when inputting sets. In contrast, the layout of truth values varied across the years. This skill was not a part of the assessment in CAA as students were given this part of truth tables. Based on this observation, a possible conclusion one can make is that it is not enough to **see** something, even multiple number of times, in order to improve. The key to an improvement is to **do it yourself**.

10.3. Results comparison by topic

Indicators can show only that students have learned one skill, e.g. a simple syntax or notation, but this does not guarantee that they will be able to answer questions that require more synthesis. To examine this we look at the marks awarded for each question/sub-question on a topic-by-topic basis. Figure 10.3 presents the year-by-year information on the percentage of marks scored in each topic. To allow for comparison, they have been calculated by weighting marks scored for each question/sub-question (not whole question) according to the marks available. This is because not all the questions were worth the same number of marks.

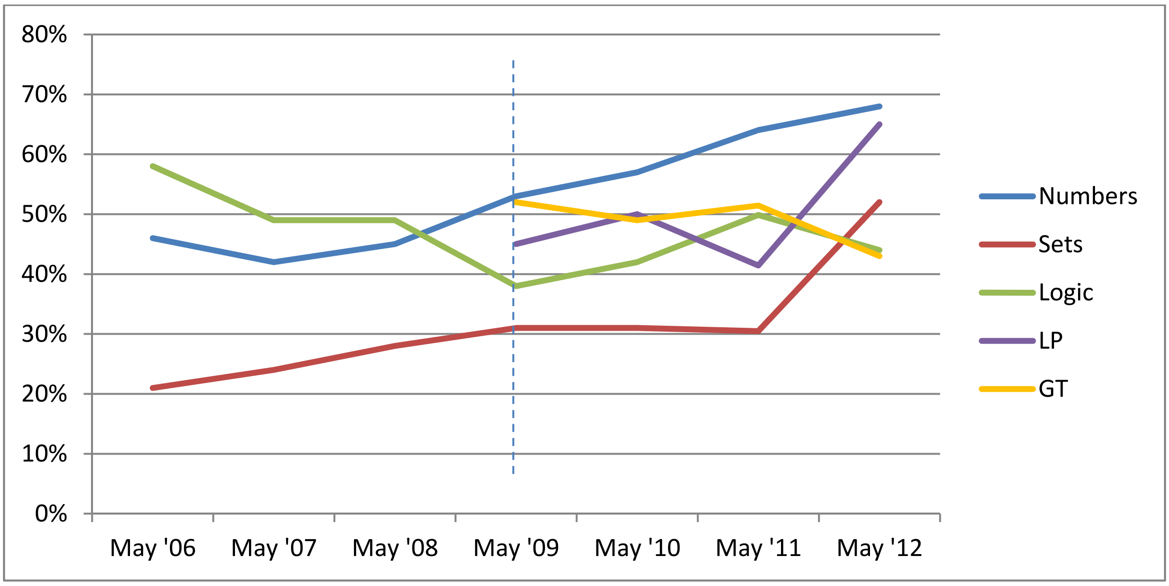


Figure 10.3: A year-by-year comparison of the % of marks scored in a topic out of the marks available for that topic. Key: LP – linear programming, GT – graph theory. In the 2008/09 academic year a new teacher took over and CAA was introduced for all topics (the dashed vertical line indicates this).

Not much can be concluded about Linear Programming (LP) and Graph Theory (GT), as they were introduced in the academic year 2008/09 and so there are no previous results to compare with. However, results for the other three topics can be compared. Sets and Numbers show an increase in average mark.

- Numbers shows a steady and continuous rise in marks achieved by students. Before students had access to CAA their result varied between 42-46%, whereas after the introduction of the software the exam results were increasing every year taking values of 53-68%. Moreover, the data from the first three analysed years shows that the performance for numbers was ranked in the middle. In May 2009 this topic had the highest success rate in comparison to other topics and this trend continued reaching a high of 68% in May 2012.

- The performance in sets is more modest. Students were scoring very few marks with the lowest of 21% in May 2006 and were clearly finding it rather hard. However, marks throughout the years prove to be going in the right direction. In May 2012 the average mark reached 52% and outperformed the topics of logic and graph theory.

Logic is different: CAA does not appear to be of benefit, or is even making things worse. Before students were using CAA, their performance in logic was the best in comparison to other topics. In May 2009 it dropped and stayed at a low level until 2012 when it only outperformed graph theory by 1%. In order to try to find a reason for that, exam results were put against the average number of times questions were answered on CAA for each topic in years 2009-2012 (see Table 10.3). Table 10.3 shows that changes in students' engagement with the logic questions was comparable with that of other topics and does not explain the drop in marks scored. Moreover, in every year it was always ranked as second or third topic that students were likely to score on. However, when looking only at numbers, sets and logic and comparing the exam results with the engagement with CAA, clear relationships are seen. Numbers were most popular in CAA all the way through, and the exam results were the highest. In years 2009-11 students answered more questions within the CAA system on logic than on sets and their performance in written assessment was rated higher. Both measurements swapped places in 2012 resulting in students scoring better on sets than on logic.

Another possible reason for a low marks achieved in logic that I decided to investigate are the exam questions themselves. In all seven years there were questions on truth tables, either for statements or to use them in order to prove equivalence or tautology. Students were also tested on the validity of arguments. What changed in the 2009 paper and carried on in the future years was that students were given no 'triangulation points'. Prior to 2009, the question format had been to **show** that two statements were equivalent and this gave them a means of checking their truth tables and rectifying any errors made. Post-2009 students were simply asked to produce a truth table. Also since 2009, students were assessed on the translation between English statements and symbolic logic and this may have cost them more time than anticipated. I recommend further investigation of students' performance on truth tables in relation to other topics on logic.

Table 10.3: For each topic, this table shows the exam results and normalised mean number of CAA questions in completed tests (as in Table 9.2). Topics are presented in the order in which they were taught.

		Numbers	Sets	Logic	Linear Programming	Graph Theory
May 06'	Exam Results	46%	21%	58%	N/A	N/A
May 07'	Exam Results	42%	24%	49%		
May 08'	Exam Results	45%	28%	49%		
May '09	Exam Results	53%	31%	38%	45%	52%
	Mean number of CAA questions	15.1	3.9	11.7	8.8	12.1
May '10	Exam Results	57%	31%	42%	50%	49%
	Mean number of CAA questions	17.6	14.6	16.1	10.8	14.6
May '11	Exam Results	64%	30%	50%	41%	51%
	Mean number of CAA questions	15.8	11.0	12.6	9.5	15.6
May '12	Exam Results	68%	52%	44%	65%	43%
	Mean number of CAA questions	16.0	14.3	12.6	8.9	7.2

Reading values from the Table 10.3 can be done directly or by rank ordering the data. Questions on sets were in many years characterised by low success in exams but this changed in 2012. This can be compared with the engagement with CAA. Throughout the years there was average interest in this topic. However, in 2012 it was second most answered question which possibly resulted in the improvement in the exam. Of equal interest is the rank ordering of the exam results and the popularity of CAA questions in graph theory. When the interest in CAA was high, students performed well in exams, but when it dropped, the same happened to the exam results. This relationship does, however, not work with linear programming (or at least not for all years).

10.4. Correlation: CAA results vs. Exam results

The relationship between CAA and exams has been investigated further and all the data and scatter graphs described in this chapter can be found in Appendix D. The performance in the CAA and the exam were analysed to determine possible associations. For this reason the results from both CAA and exams for all four years (2009 – 2012) were put together. A scatter graph was produced (Figure 10.4a) and PMCC (Product

Moment Correlation Coefficient) calculated. PMCC is "a measure of the strength of a linear association between two variables and is denoted by *r*" (Laerd Statistics, 2013). As mentioned in Chapter 9, it can take values between -1 and +1. If it is 0 then there is no correlation, the closer it is to ±1, the stronger it is. The sign depends on whether it is a positive or a negative correlation.

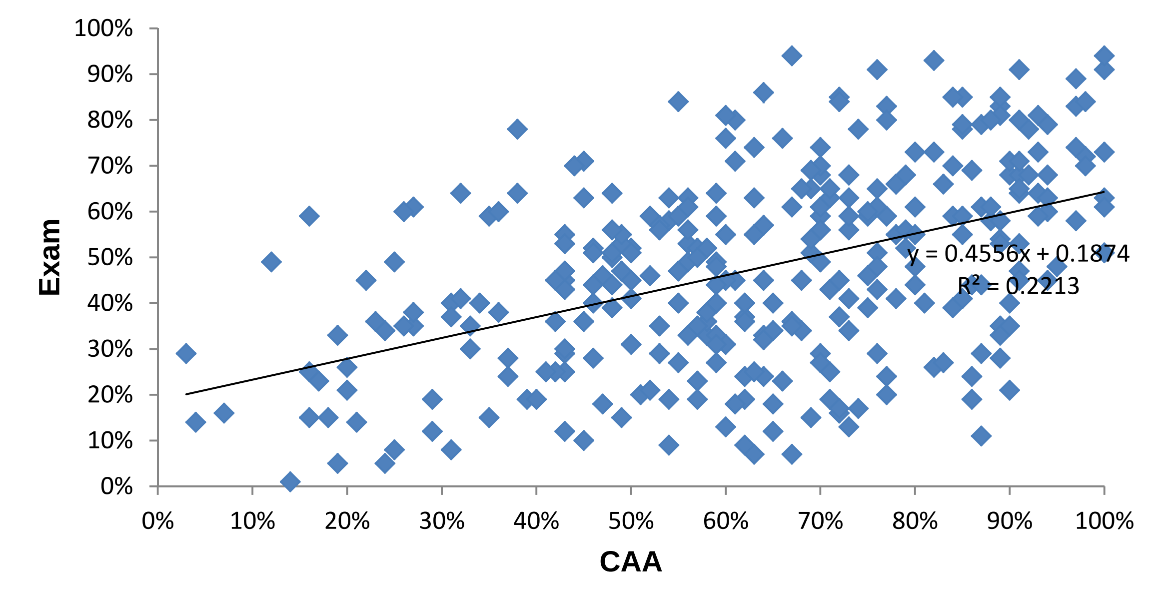
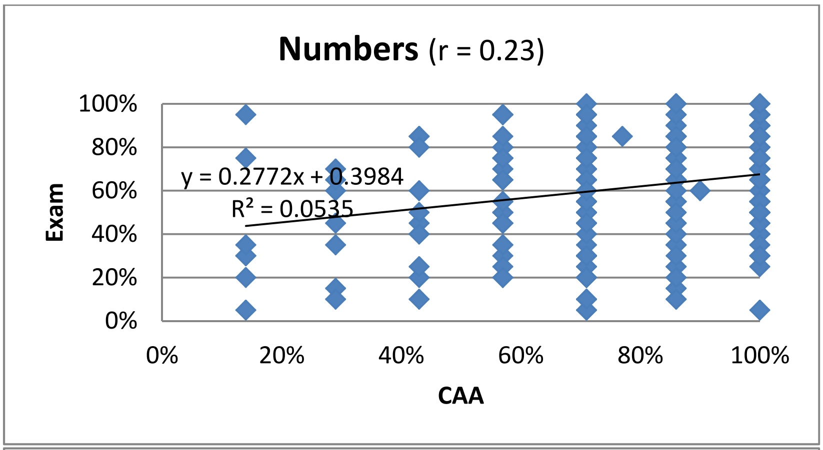


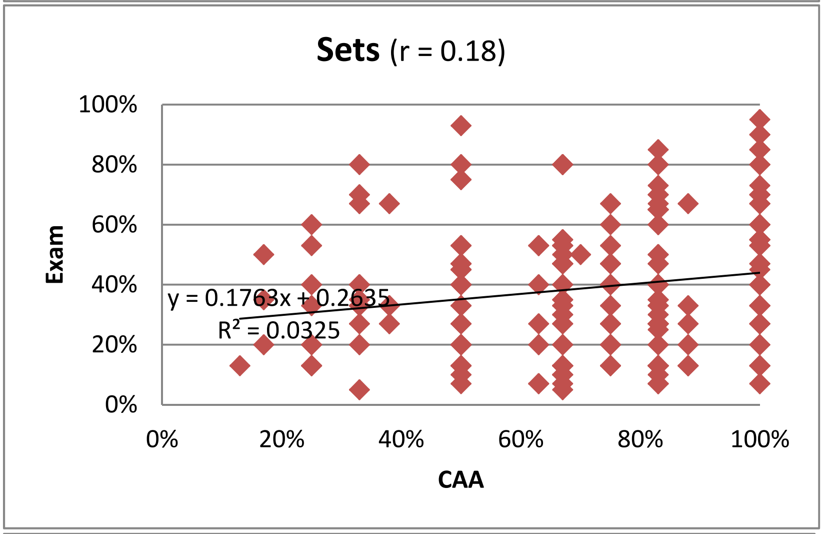
Figure 10.4a: Scatter graph showing the correlation between CAA results and exam results of the data combined for five topics in four academic years (2009 – 2012).

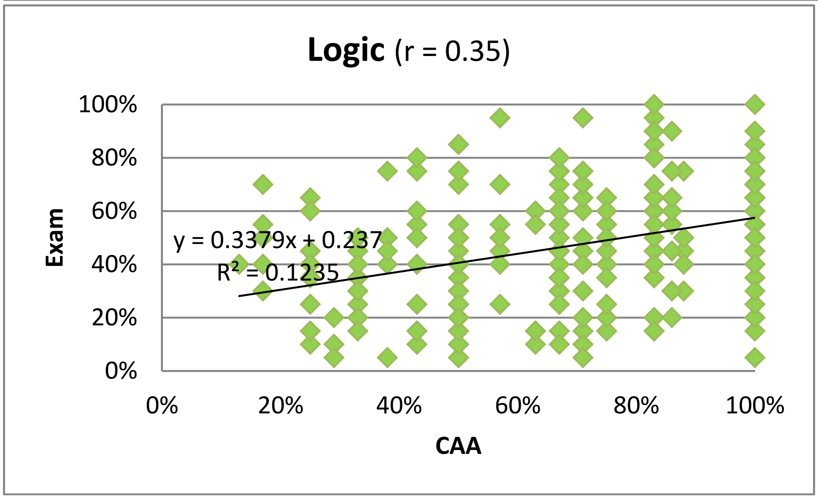
Figure 10.4a contains the information on the performance of 312 students in the CAA and in the exams. The exam result is a percentage of marks scored out of all marks possible to score (80 since best four questions were taken into account each worth 20 marks). The CAA score is an average of five tests (one for each topic and taking the best result in each topic). Figure 10.4a combines data from years 2009-2012 for all five topics. The minimum requirement for the scores to be counted was that students must have scored some marks in both assessments. As little as 1% for exam and 3% for CAA have been recorded and therefore included in the graph and to calculate the PMCC. A correlation coefficient r = 0.47 indicates that there is medium association of exam results on CAA mark. Students who got higher scores for the CAA tended to get higher exam marks. For individual years the weakest correlation, r = 0.34, was in year 2008/09, but increased throughout the years with r = 0.41 (in 2009/10) and r = 0.44 (in 2010/11) to the strongest, r = 0.68, in 2011/12. This variation does not seem to depend in any obvious way with the measure of student engagement given by the mean number of CAA questions attempted given in Table 10.3 and its cause is unidentified, see Recommendations. The scatter graphs for each year were produced and can be found in Appendix D together with all the data and above stated PMCC (r) values.

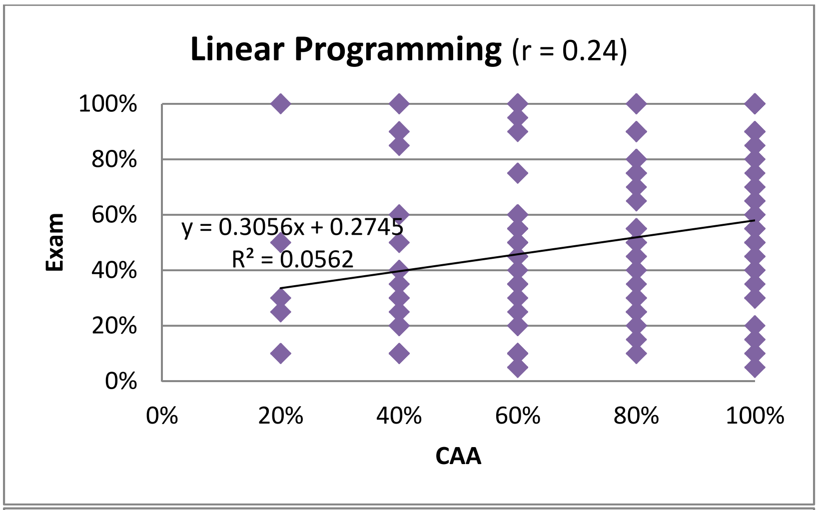
Rasila *et al.* (2010) describes a similar study that took place at Aalto University. It describes three years' worth of data for the basic course in mathematics taken by the first year students of Electrical and Telecommunications Engineering (consisting of complex numbers, matrix algebra, linear systems of equations, eigenvalues, differential and integral calculus for functions of one variable, introductory differential equations and Laplace transforms). The correlation was measured using Spearman's rank correlation coefficient (Laerd Statistics, 2013) which in contrast to Pearson's product-moment correlation coefficient uses ranked values, not the measured values. The results show a significant correlation between the score from the e-assessment (STACK) and the exam results, with the Spearman's rank correlation coefficient between 0.57 and 0.71. Rasila *et al.* (2010) also presents the correlation for Discrete Mathematics (mostly students from second to fifth year) which takes value of 0.73. Furthermore, Spearman's rank correlations of the scores of traditional exercises with exam scores take values between 0.49 and 0.69. Since these authors quote Spearman's rank correlation rather than the more informative Pearson's product-moment correlation used here, no direct comparison is possible beyond noting the similarity of their values with those presented here.

The correlation between CAA and exam results for each topic is also of interest. To find that, values for both assessments are required. However, since not all the questions were compulsory in the exam, one cannot simply assume that the question was not answered due to lack of knowledge and therefore assign 0% to it. For this reason, if either of the two marks was missing, the other one was deleted. With data amended in this way, the PMCC calculated shows that in all but the last year there was no correlation in Sets. This possibly explains, to some extent, the poor student performance discussed in Section 10.3. The most consistent was for graph theory with r varying between 0.37 and 0.43. There is, however, no consistency in other topics. The PMCC for combined years for Numbers is 0.23, Sets 0.18, Logic 0.35, Linear Programming 0.24 and 0.38 for Graph Theory. This means that students who got a higher CAA result tended to get higher exam marks. The trends of the correlation between CAA results and exam results for the five topics and the four years (2009-12) can be found in the scatter graphs in Figure 10.4b.


Numbers (r = 0.23)
100%
80%
60%
40%
20%
0%
Exam
y = 0.2772x + 0.3984
R² = 0.0535
0%
20%
40%
60%
80%
100%
CAA


Sets (r = 0.18)
100%
80%
60%
40%
20%
0%
Exam
y = 0.1763x + 0.2635
R² = 0.0325
0%
20%
40%
60%
80%
100%
CAA


Logic (r = 0.35)
100%
80%
60%
40%
20%
0%
Exam
y = 0.3379x + 0.237
R² = 0.1235
0%
20%
40%
60%
80%
100%
CAA



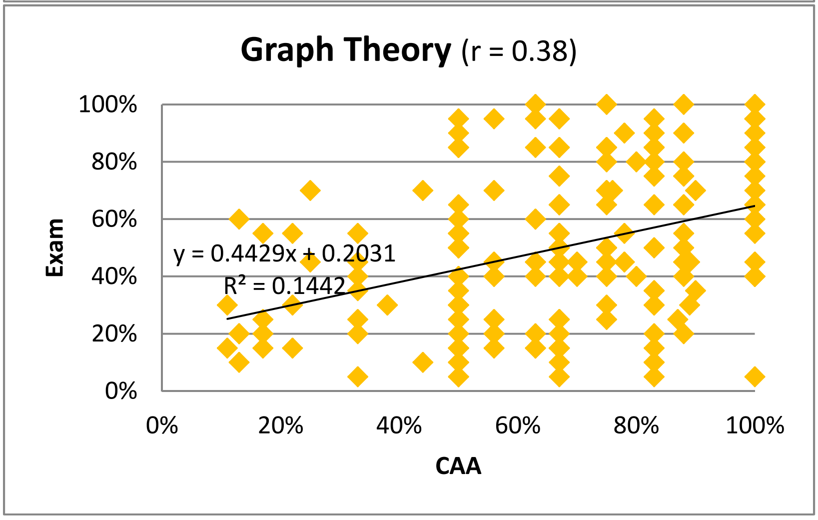


Figure 10.4b: Scatter graphs showing the correlation between CAA results and exam results for five topics (numbers, sets, logic, linear programming, graph theory) combining four academic years (2009 – 2012). Note that the CAA marking scheme allowed only a discrete set of marks to be awarded and this gives the stacked marks in the above.

The PMCC's for each year of combined data showed a year-on-year increase of the correlation between scores achieved in CAA and those of exams. These values were relatively higher compared to those of each topic, meaning that when the data is combined there is a higher correlation. As a result, the PMCC was calculated for each topic in each year and Table 10.4 presents these correlations, together with those showed above.

Table 10.4: PMCC values for: each of the five topics in each of four academic years, combined years for each topic, combined topics for each year, combined topics and years.

	Numbers	Sets	Logic	LP	GT	combined topics
2008-09	0.25	-0.05	0.21	0.18	0.43	0.34
2009-10	-0.10	0.05	0.21	0.28	0.41	0.41
2010-11	0.38	0.06	0.41	0.12	0.37	0.44
2011-12	0.25	0.40	0.38	0.44	0.37	0.68
combined years	0.23	0.18	0.35	0.24	0.38	0.47

To allow better picture of all the numbers this data is presented graphically in Figure 10.4c. As one can see, there is a clear increase of the correlation for the combined topics. A possible interpretation of the data in Table 10.4 is that the CAA, as it progressively gets edited over the years, is improving as an assessment method. However, note that only a small percentage of the questions were in fact edited and then only to correct rather trivial points of English. A more substantive change, and hence a possible confounding influence, was the addition of the questions developed later in this thesis, but this would affect only the Sets and Graph Theory results. However, from the data in Table 10.4, no clear picture emerges. This is not surprising given that the mean exam marks shown in Figure 10.3 do not show any consistent improvements that could be attributed to the addition of new questions either. When looking at each topic separately, the PMCC values vary. Appendix D includes the scatter graphs for twenty cases (five topics for each of four years). However, there is no clear picture that students were performing better or worse in the exams in comparison to their CAA scores for each topic as time progresses.

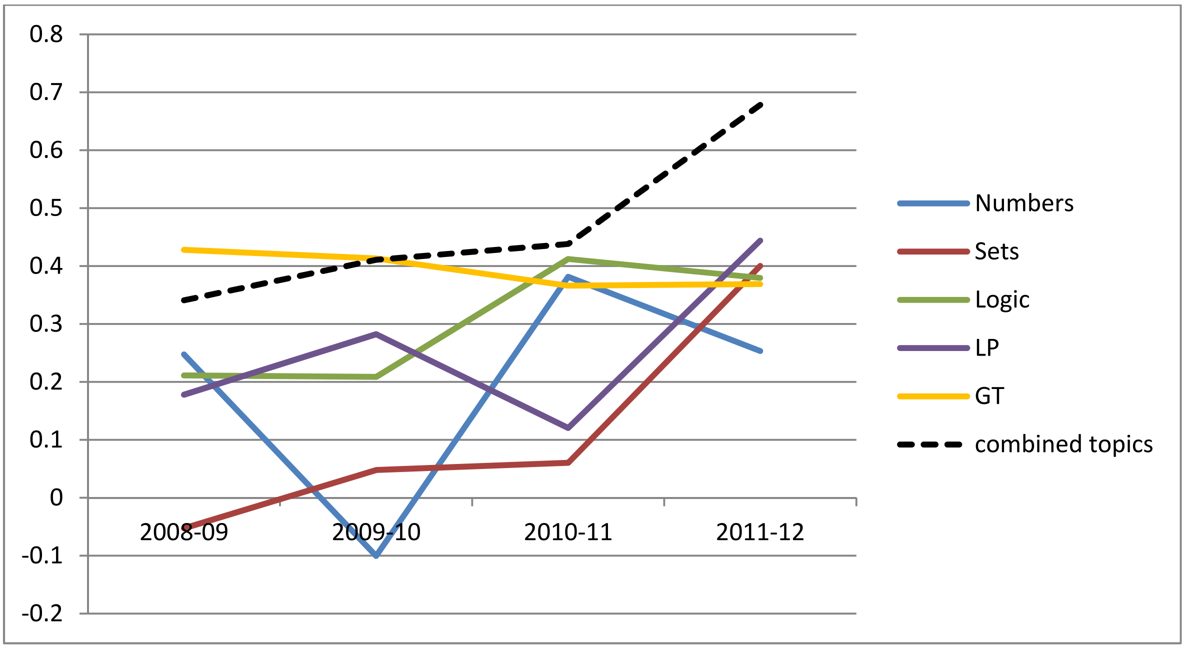


Figure 10.4c: A year-by-year comparison of the Product Moment Correlation Coefficient (PMCC) between CAA and exam results for five topics (numbers, sets, logic, linear programming, graph theory) as well as combined topics.

11. Conclusions

11.1. General overview

The research question: **How might CAA influence teaching and learning?** is obviously very broad and may not have a consistent answer across all, or even most, disciplines. It becomes much more feasible to say something useful if one limits the question to the mathematical parts of a wide range of STEM subjects and some social sciences, for example Economics, which have substantial mathematical content. This thesis has looked at a specific area of mathematics, namely discrete mathematics, and its delivery to a specific target group, namely Foundations of IT students at Brunel University. The module syllabus for these students naturally falls into five distinct parts (Numbers, Sets, Logic, Linear Programming and Graph Theory). A useful feature of this module is that it can be taken from scratch with very few prerequisites; indeed beyond some basic arithmetic and algebra, and the equation of a straight line (as specified in Chapters 3-7) nothing is assumed of the students. A few students will have taken A-level modules in Discrete Mathematics, usually just the Edexcel D1 module or equivalent, and some will have seen numbers in A level Computing. However, for the purposes of this study, we have ignored these confounding influences believing them to be of little significance (this would certainly not be the case for CAA in elementary algebra for example). Thus we assume that the study starts from a 'clean slate' and that any effects measured are due to the controlled teaching and (to a lesser extent) learning whilst students are at Brunel University.

An obvious question is then how can we generalise the conclusions below to a wider range of mathematical topics and a wider cohort? This question comprises at least three distinct parts:

a) Can we apply the techniques of CAA developed here more widely?
b) Is the implementation of CAA likely to be as effective in other low-level university modules, or even at school level?
c) How can CAA influence teaching practice?

In answer to a) it should be noted that whilst the pedagogy of question design is influenced by the question content, the technology is not. A wide range of elements (equations, tables, graphs etc.) and a variety of question types were needed to provide students with a useful resource and these are widely used elsewhere. The pedagogy of setting questions is also 'exportable' beyond the 'obvious' issues of short clear question specification, an uncluttered screen layout and very full feedback screens, to more subtle issues to do with the specification assumed and tested skills for each question. In particular, the questions should not necessarily ask for the 'answer' (which students may achieve via a variety of, possibly heuristic, ways) but rather test the implementation of a method or algorithm. For example, students may be able to 'see' the minimal spanning tree, without being able to use Prim's or Kruskal's algorithms (and we would want control of which algorithm was to be used in any case). So the lessons learned here in the development of the questions are likely to be of use much more widely.

In answer to b), the present study presents a rather large study over many years and with a cohort of around 100 students per year, to underpin the claim that students perceive CAA positively and that it influences their learning (or at least their ability to do related exam questions). A schema for embedding CAA in the curriculum has been described (we chose to take the best of the first five attempts at any test): although in HE for mostly summative use, this schema can be widely adopted, other schema are certainly possible and used elsewhere (e.g. practice tests followed by an invigilated test for marks). Some of these might be more suitable for high-stakes assessments and or use in schools. However embedded, the effects we measured and are likely using other schema, are often not that marked, but there was an overall trend for improvement of student engagement and achievement when CAA was introduced and this is likely to apply to other modules/student cohorts. Note that the cohort is foundations students who are usually rather weak at mathematics and sometimes maths-phobic. CAA seems to offer them a ‘safe environment’ in which to make mistakes and hence learn, see below. The thesis also goes beyond this simple questionnaire analysis, which relates to their perceptions, to a detailed study of the answer files and examination scripts. It is worth remarking that the CAA system used facilitates detailed analysis of question facility and discrimination in a way that would be very onerous otherwise and hence might be used to replace the largely anecdotal feelings of experienced teachers. A great deal can be learned from both statistical measures: facility should influence teaching, spending more or less time on specific topics/techniques and discrimination should influence the setting effective assessments. This, at least in part, answers c) above in the affirmative.

11.2. Question design and coding issues

We now address the research question: **Is it feasible to write questions in the areas of discrete and decision mathematics?** Again the answer is yes, although one needs to acknowledge the limitations of objective questions *per se* and their computer delivery in particular. As far as the pedagogy is concerned, the design and range of the questions demonstrates that valid questions can be set for these subtopics. Certainly, the predominantly numerical-input type questions provide an acid test for whether or not a student has mastered a particular skill or skills since a mistake at any stage will result in a zero mark. Possibly some questions with intermediate stages are needed, but the data indicates that unsuccessful students can learn from the feedback screens enough to achieve in subsequent tests.

The main message from the question design process is that the question type and design must be carefully chosen to ensure as far as possible that the question actually tests what the author wants to test and that students are constrained in the way that they answer it. One needs to be fully aware of how much information is given in the question and in its format e.g. the number of input boxes etc.

The research question**: What are the most difficult and most discriminating questions?** is only likely to be usefully answered by adding ‘**within a particular topic**’. We address each topic in turn below.

11.2.1. Numbers

We can conclude that setting effective questions is generally possible for the Numbers topic, although not always with certainty. Noteworthy points are:

- For prime factorisation, valid questions necessitate ‘reverse engineering’ the question, so that the randomly-chosen prime factors have properties that are under control. Basically this means that the prime factors should not be too large (<230) and that repeated factors are/are not allowed as required. It is desirable to have similar questions in different types, for example for prime factorisation the XNI type shows the correct number of input boxes (dynamically) whereas the more difficult WI question simply asks for a non-decreasing sequence of numbers, so students do not know how many factors there are and also have to understand ‘non-decreasing’. Related to this are questions on HCF and LCM: here the school approach of completely factorising the input numbers means that they have to be considerably smaller (typically <100) than the inputs where it is assumed that students will use Euclid’s algorithm (typically 6 to 9 digits long). (Complete factorisation would be possible of course, but this is highly inefficient and if chosen by the student, would require far more work.) The question is designed so that at several passes through Euclid’s algorithm are required, never just one. There is also a question on a simple primality test; although not hard, this question cannot be done directly since the number is not given; rather the question is designed to test a deeper understanding of factorisation. Underpinning these pedagogic considerations is effective use of screen layout that uses tables and diagrams, both of which change size dynamically according to the factors of algorithm being used.

- Modular arithmetic proved rather hard for students since input numbers were generally not given and this then removed the ‘crutch’ of their calculator. The questions deliberately tested the students ability to think algebraically since that was the skill that was to be tested.

- Questions on non-decimal arithmetic all required students to implement some sort of algorithm. All input numbers were given and a numerical input was required. Straightforward calculations proved easy enough, but finding the complements (rather than doing the subtraction) defeated many students. It is of course a worry that students may have chosen to convert numbers back into decimals, then done the arithmetic and then converted back to the specified base; worse still some calculators can work natively in binary, octal and hexadecimal so then the question would have been worthless. One can conclude that although these questions were popular and students appeared to learn from the feedback, some should probably be re-designed to force students to use a specific number base (by hand).

- Questions on scientific notation and significant figures were relatively straightforward to author and code. Of note are the questions on the range of possible answers given arithmetical operations with two numbers given to a specified precision; these tested students' understanding at a deeper level than simply using a calculator and rounding.

11.2.2. Sets

We can conclude that setting effective questions is generally possible for Sets, although not at a more advanced level requiring proof. MathML was capable of all notation needed, whereas the html overbar and cup and cap did not display correctly on some browsers and were hence avoided. Venn diagrams were displayed clearly using SVG where appropriate. Other noteworthy points are:

- For notation and enumeration of sets resulting from set operations with two sets, the WI questions are effective and do not give very much information away since an input string is required. The format of the string needs to be precisely specified by the student, otherwise they will be marked incorrect. Anticipated complaints about this did not, in fact, happen often: students appeared to learn the importance of correct notation from this very unforgiving marking scheme. This is valid since correct notation is a significant part of what is being tested in these questions.

- The TFU questions on equality, subsets, set identities etc. were designed not to test students' ability to apply a technique or algorithm, but rather to test their understanding. From the data presented in Chapter 9 which showed medium values of facility and discrimination, we can conclude they do this effectively and probably make students 'stop and think' more. The same comment applies to questions on cardinality.

- Questions on power sets, Cartesian products, and partitions use a range of types, some asking for an answer and some for a specified property of an answer (such as its cardinality). Again the data presented in Chapter 9 shows this to be desirable for power sets (Cartesian product and partitions questions were not available for all five years and hence are excluded).

- Questions on Venn diagrams are of WI type. Questions include Venn diagrams with two or three overlapping circles. Each region of diagram is numerated. Questions test understanding of representing symbolic representation of set operations graphically by stating the numbers corresponding to regions that should be shaded in. There is no information available on questions' facility and discrimination since they were not available for testing at the time of this study.

- The counting principle questions are reverse-engineered from positive numbers in each region of the Venn diagram (random numbers could result with a negative number of elements in some places, therefore the constraints are needed). Questions with two and three sets are available. The analysis of answer files generally shows high facility of the questions. The results of the correlation are not so consistent and vary between years.

11.2.3. Logic

At the elementary level presented here, Chapter 9 shows that effective questions can be written for Logic. From the authoring process, the following points have arisen:

- Questions on truth tables are of the WI type with clearly specified format of the answers (a string of letters T or F, separated by commas and with no spaces). The requirement on no spaces was causing several complaints from students that they were wrongly marked. It forced the change in the coding and inclusion of the *trimall()* function deleting all the spaces in the submitted answers. Students are provided with the table completed by the statement and truth values for involved propositions and are tested on finding truth values of the statement. These questions have varying facility (statements containing NAND, NOR, XOR, converse had a lower mean score) and discrimination.

- The MC questions on translations were designed to test students' understanding of the logical meaning of English words and the ability to translate symbolic representation into English. Answer files suggest that these questions are easy and well discriminating.
- Questions on applications are of multiple-response type and are dressed in the theme of turning playing cards. Students are tested on applying their knowledge on conditional statement and directly related to it converse, inverse, and contrapositive statements. These questions were correctly answered by 30-50% of students and it is assumed that broad number of skills makes these questions hard despite the list of answers for selection. It is tempting to speculate that some

students did not understand the English involved in the question and hence did not know what was required of them. As a result, these questions discriminate relatively well.

- Questions on connectives are also of MC type. They test the understanding of the conjunction and disjunction in algebraic equations. Table 9.2.3 shows rather high facility and positive but not as high discrimination as some other questions on logic.

11.2.4. Linear programming

We can conclude that setting effective questions is generally possible for Linear Programming. MathML was capable of all notation needed to display inequalities for constraints and equalities for objective functions. Graphs showing the constraints as well as the objective function displayed clearly using SVG. Other noteworthy points are:

- Questions on feasible region are of the MC type. All the coefficients involved in formulating the inequalities are under control and later used to generate the relevant graph. Throughout all the possible answers the coefficients are the same but inequality signs vary. This is because questions test whether students know which half-plane of the graph represent which inequality (below or above the equation line). The data presented in Chapter 9 shows high values of facility and discrimination.

- The RNI questions on optimisation involved applying sequential tasks correctly. Students had to formulate constraints and objective function and then understand the roles of their slopes in finding the optimal point. Students' performance reflected the complexity of these questions through low mean score. On the other hand, high correlation shows that questions are highly discriminating.

11.2.5. Graph theory

It is possible to set effective questions for Graph Theory, although, as in other topics, not at a more advanced level requiring proof. Graphs/digraphs were displayed clearly using SVG where appropriate. Due to some problems with a display of weights in networks, students were provided with the network matrices which displayed correctly. Other noteworthy points are:

- Questions on adjacency matrix are of the MC and RWI type. The size of the graphs is controlled by the defined constraints on the number of vertices (between 4 and 9) but also their degrees (between 0 and 2). Questions test the ability of finding graphical representation of adjacency matrices or vice versa. The data in Chapter 9 shows that students were performing well on these questions but this may be due to the fact that students mainly had to compare the given information.

- Questions on degrees, but also on edge and vertex sets test students' general understanding of the concepts of graphs and their ability on reading graphs by defining key properties (degree sequence, indegrees, outdegrees, edge and vertex sets). In the case on NI questions there are no issues with marking. To avoid those for WI type questions the format of the answer is specified. High mean scores for these questions show good understanding of the basis of the topic.

- Some questions on (minimum) spanning trees required students to implement either Prim's or Kruskal's algorithm whereas in others it was possible to 'spot' a correct answer without implementing any algorithm. The correct answer can be achieved in more than one way. To control that, questions asking for middle steps when implementing an algorithm were coded. The main challenge in the questions design was the feedback. To allow students learn from it it was important to include in it more than just the definition of each algorithm. The feedback includes step-by-step explanation of the relevant algorithm. It shows graphically how the minimum spanning tree builds up and it also states any edges that are rejected in that process. These questions proved to cause some problems to students and were of medium discrimination.

11.3. Student engagement and performance

We now address the research question: **How does the formative feedback affect the results students achieve in exams?** As above we can only answer this question usefully by adding '**within a particular topic**'. We therefore address each topic in turn below.

11.3.1. Numbers

Students' positive feedback about CAA is confirmed by their engagement. Taking the data from five years (2009-13) into account, on average they took 2.24 tests on the Numbers topic. In 2010 they were most engaged with 2.5 tests completed and least in 2013 solving 2 tests. Since the number of questions varied across the years and topics the data was normalised and in 2013 students solved on average 17.9 questions while in 2009 15.1 questions (16.5 questions on average throughout five years).

The positive correlation with exam performances indicates that the questions work well. The exam results reflect those of CAA in some extent. The scatter graph (Figure 10.4b) was produced for the results of CAA and exams combining the data from four years and the correlation (PMCC denoted r) calculated. For Numbers r = 0.23 and is the second weakest value when comparing the results for five topics.

The analysis of exam scripts with respect to two indicators suggests that CAA has a positive impact on the exams' performance. The percentage of students being able to write remainders as well as convert numbers between different bases increased after CAA was introduced. Moreover, exam marks increased by 11-22% (depending which of the seven years of data was taken into account).

11.3.2. Sets

For sets, the engagement dropped slightly but still stayed at a high level. Combining the data from five years, on average 2.19 questions were solved (only 2 tests in 2009 and 2.4 in both 2010 and 2012). In 2009 there were not many questions and one test included questions on both sets and logic. This resulted in only 3.9 questions being solved in that year. In 2013 there was a significant increase in the engagement and students solved on average 21 questions increasing the average number of questions for five years combined into 13.

The correlation between CAA and exam results for the topic of sets is positive but smaller than for other topics. The scatter graph (Figure 10.4b) was produced for the results of CAA and exams combining the data from four years and the correlation (PMCC denoted r) calculated. For sets r = 0.18 and is the weakest value when comparing the results for five topics.

Exams scripts were analysed with respect to the correct notation of sets, i.e. usage of curly brackets and it has been proved that there was a significant increase with this. Exam results indicate that students find the topic difficult as only in the last year the marks scored increased enough so the topic was not the least successful that was the case in past years.

11.3.3. Logic

As the year was progressing, the number of tests solved continued on dropping and 2.04 tests were solved on logic. Students were least engaged in 2013 solving only 1.7 tests and most engaged in 2010 solving 2.7 tests. As logic was a part of the test in 2009, in that year only 11.7 questions were solved and 16.1 questions in 2010. These values were smallest and biggest entries for logic.

The correlation between CAA and exam results is positive and has the second highest value of all topics proving that the performance at the exam reflects the results students achieve in the CAA. For logic r = 0.35 and is the second highest correlation when comparing the results for five topics.

The analysis of exam scripts with respect to one indicator does not show that students implemented in the exam papers what they have seen on the screen of the CAA. In the first two years (2009 and 2010) of using CAA the number of students laying out the columns of truth values in a specific way decreased in comparison to the years when students did not have access to the CAA, but increased in the following two years exceeding the best performance prior to 2009. The performance at the exam also decreased in 2009 and stayed at a low level throughout four years of using CAA.

11.3.4. Linear programming

Students were least engaged with linear programming. On average, students solved 9.3 questions on this topic (1.86 tests) when the data was combined from five years. The highest number of questions solved was 10.8 (2.2 tests) and occurred in 2010, while the lowest was 8.3 (1.7 tests) in 2013.

The correlation between CAA and exam results is positive and comparable to the one of numbers. The data shows that it was one of the least engaging topics in CAA and reflected in rather weak performance in the exams, with an exception of year 2012. For linear programming r = 0.24.

Since the topic was introduced to the course in 2009 in line with CAA the exam scripts were not analysed with respect to any indicators. Moreover, it is impossible to compare the students' performance from before the CAA and after. However, it is worth mentioning that students did relatively well in the exams, especially in 2012 when there was a significant improvement and an average exam mark reached 65%.

11.3.5. Graph theory

The number of questions solved a year is a normalised data and therefore reflects true engagement. The number of questions solved increased in relation to linear programming and the average per students was 12.5 (but only 1.73 tests). The CAA was least popular in 2012 (7.2 questions) and most popular in 2011 (15.6 questions). In 2012 students solved, on average, only 0.8 tests and 2.4 tests in 2010.

The correlation between CAA and exam results is positive and the highest of all topics. It has been observed that after a decrease in the engagement in linear programming there was an increase in the number of student accessing the CAA for graph theory questions with an exception of 2012. Then the engagement dropped further what resulted in graph theory being a compulsory topic in the exam in 2013. For graph theory $r = 0.38$.

For the same reason as linear programming, graph theory questions were not analysed for any indicators. In years 2009-2011, students were performing well in the exam however with the increase of marks in other topics and a decrease in graph theory, it became the least successful topic in 2012.

11.4. Validity of the methodology

We now address the research question:

- **Can we establish the methodology on how effective the questions are?**

Certainly a practical methodology has been used in this thesis, based on the work of Gill (2007). This comprises a questionnaire to measure students' perceptions, and an analysis of CAA answer files and exam scripts. The CAA answer files are analysed at a fairly high-level, but more could be done to explore student inputs and hence begin to answer the question of precisely **why** a particular question is hard or discriminates well. To keep the textual analysis at a manageable level, exam scripts were handled at the almost trivial level of indicators. Again more could be done, but it seems essential to have some sort of categorisation of the procedures and errors students write, with associated metadata. This is a very hard task and is not yet available. So we can establish a practical methodology, but make no claim that it is the only possible one or even a good one amongst other possibilities. Ideally these should be compared, see Section 10.3.

The questionnaire, including a mixture of closed and open questions, was distributed in one year and returned by only 30% of students. For the data to be more reliable it should be repeated. Chapter 8 presents the findings and also mentions the results of Economics students with a bigger return rate and higher mathematical skills. However, concentrating on the FoIT students some conclusions can be drawn. The intention of CAA is that students use it not only as an assessment but also as a learning tool and this was confirmed by 79% of students responses. Moreover, the same proportion of participants stated that they would use CAA even if it was not compulsory (18% as much as now, 32% once or twice, 29% only for the feedback). 72% of students said they are planning on using the system for revision. This, however, could not be reviewed since the questionnaires were anonymous. Even if we would check the statistics we would not be able to say whether the students who revisited the CAA were those taking part in survey. The cross-tabulations on the questionnaire resulted in the statement that men favour CAA in mathematics module. The open questions revealed that students appreciate the format of the tests (5 attempts, feedback, no time limit, possible collaborative work, etc) while others would still like more attempts or were not happy about strict marking scheme, too many questions and freezing network. Neither in the questionnaire, nor in any other way, were there any topic-specific complaints. Some students pointed out that some questions lack in feedback, which, by implication, means that they do appreciate and learn from the feedback in other questions. As a result all the questions now have feedback. The surprising finding questioning the accuracy of students' perceptions was that those who see CAA as an assessment and learning material (as opposed to assessment **only**) are less likely to access the tests if not compulsory.

The aim of the questionnaire was to hear students' direct opinion about the CAA. Although Economics students were not part of this study, the purpose of showing their opinions was to provide a wider scope and to see if responses are largely independent of the topic content (whether it is calculus or graph theory). The high failure rate (30-50%) of FoIT students and low level of motivation shown in the poor lecture attendance resulted in the return of only 30 surveys. Therefore, the comparison with Economics students was an attempt to quantify to some extent whether or not weak students feel more or less negative about using CAA. This, however, was found at the comparable level with no negative feedback. Since the questionnaire was anonymous we believe that given answers were honest and show a real picture of students' perspectives. For the suggestions on extending the questionnaire to other groups of students and by more questions see Recommendations. Students' negative comments about the lack of feedback in some questions resulted in it being included where such was missing.

As described above, answer files provided a wide range of hard information. Questions were classified with respect to their facility and discrimination. This allowed informing teaching and questions for improvement as discussed in Recommendations with respect to particular topic. The measure of engagement (number of answered tests/questions) with all five topics showed trends and proved that it is higher at the beginning of the year, and then drops to increase at the end. Moreover, the available data spans five years, a total of many thousands of questions (and for single questions hundreds or tens) were analysed, being generated by different student cohorts of typically 90-110 students per cohort.

The exam scripts were analysed by the author and spanned seven years' worth of data. Four easily-identifiable indicators were chosen to avoid any speculations in the process of classification. This could take place if one would like to measure the understanding of the concept instead of specific skills/abilities. Another reason for choosing writing remainders, converting numbers between different bases, using curly brackets for sets and laying out truth tables in a specific way, was that they were possible to define across years in the exam scripts but also in the CAA. The exam results of particular topics prior to CAA and after it was introduced were compared. These two ways of looking at the scripts was adequate for answering the research questions of the effectiveness of the CAA questions and the possible effect on the exam performance.

11.5. Recommendations

There are at least two aspects that pertain to the implementation of CAA within a module or course: those directly affecting the students' learning by proper design of questions and feedback, and those affecting the teaching staff (and hence indirectly the students) by informing the way the material is delivered. Firstly the mundane, but important, schema for embedding CAA within a module (assessment weight, availability, group work allowed, invigilated or not, multiple attempts allowed etc), should be re-examined with control groups to assess the effects of different schema on student learning. This would require proper experimental design and statistical analysis and, very likely, ethics clearance. At a deeper level, the style and content of the teaching of the topics for discrete mathematics, and more broadly, should be influenced by a detailed determination of the most common mistakes students make in both exam scripts and in the CAA by investigating what their inputs were. Making sense of this mass of detailed data is likely to be possible only within some sort of broad framework or categorisation of the errors made. This taxonomy needs to contain detailed enough metadata to capture the important information, yet be general enough to reduce the data to a manageable size so that common errors span several questions or even topics, e.g. incorrect commutation of operations. If this is possible, it would allow multiple choice or responsive NI or WI questions to be redesigned on a rational, rather than an *ad hoc*, basis. This could also allow the feedback to be specifically directed to a submitted answer in the existing questions. Beyond that, understanding how students go wrong could also influence other teaching material, although some textbooks e.g. 'Decision Mathematics 1' by Hebborn (2000) do already stress likely errors.

Taking each topic in turn, the following topic-specific recommendations are made:

More teaching time should be spent on complements, significant figures and possibly on modular arithmetic. For example, modular arithmetic, simpler questions on finding the modulus of a number, e.g. 7 mod 5, should be included in the teaching and assessment to allow practice and preparation for what is found in the data analysis of the answer files to be difficult question: 'If x mod 65 = 31 and y mod 65 = 36, what is $(x+y)$ mod 65?'. Similarly, knowledge of significant figures should not be assumed. According to the lecturer, quite a lot of time was spent on complements, so simply spending more time on the topic is not likely to be successful; other approaches need to be tried, but beyond more practice with the CAA or paper-based exercises, it is not clear what these might be.

It should be researched why students find the questions on counting principle difficult and therefore are unsuccessful with them in CAA. One way to investigate this is to analyse the responses students provided when answering the questions. Secondly, they could be asked to provide their workings on paper or conversations could be carried out where students explain their understanding of problems. More time should be dedicated to teach counting principle together with the definitions of integers and absolute value. Moreover, drag-drop type questions on finding the set-builder notation would allow testing this skill. The sets could be given explicitly or written in English and students would have symbols such as: |, <, >, ≤, Ø, ≥, ∩, ∪, ∊, ∉, ℤ, ℕ, ℚ, ℝ, ∧, ∨ available to drag and drop into an input box. However, as the set described in words can be written as a set-builder in one way, but a set given explicitly could be written in more than one way, this makes designing questions for the latter more challenging. This would probably involve external software checking the correctness of the set with the one from the question. When working with numbers it is possible to get one answer in more than one way (e.g. $2 + 2 = 4$, $2^2 = 4$, $10 - 6 = 4$, etc.) and so is possible with sets. Therefore, the use of a symbolic manipulator should be considered to go beyond simple numerical input.

More teaching time should be spent on applications of conditional, converse, inverse, and contrapositive statements. Additional questions could be coded to test students' understanding of those statement (not the applications of them) and then also different to the card turning theme on applying that knowledge. The NAND, XOR, NOR, and back arrow for implication should be highlighted more to students. Similarly to what was suggested for sets, here the drag-drop questions for translation of English statements to symbolic logic could be designed as the MC questions are found to be very easy and limit the possible answers. Also symbolic manipulator could be used to ask students for a possible statement giving certain truth values. As suggested in Section 5.1.1 questions on equivalent relationships of single operators could also be coded.

As suggested in Section 9.2.4 more teaching time should be dedicated to optimisation in order to understand what makes the questions difficult. It is assumed that it is either due to the need of sequential tasks (formulating the feasible region and objective function, understanding the roles of their slopes in order to find and evaluate the optimal point) or having problems with the evaluation of the objective function at each vertex of the feasible region. Similarly to the issues with counting principle it could be worth collecting students working to the problems on optimisation questions. Questions testing middle steps that were listed for sequential tasks could be of benefit to students but also could help to determine where the possible misunderstandings are. Instead of giving students possible answers for constraints, symbolic manipulator could allow students to type these by themselves and display them graphically.

More teaching time should be spent on (minimum) spanning trees and algorithms associated with the topic. It should be stressed to students that numbers in networks indicate the weights of edges not their amount. Since the questions on adjacency matrices were found to be very easy it is worth designing questions which would require students to produce the matrix by themselves instead of indicating one of few options for the answer. This could be a multiple numerical input question displayed as a table (possibly only with the outside borders displayed). Questions on shortest path using Dijkstra's algorithm are coded but they require the step-by-step feedback to be finalised. The question includes the information about changes to the weights on each edge but due to the complexity it was not presented graphically and still needs doing. Existing questions cover most of the level one of computer science material, but more should be written for the level two Algorithms and their Applications, for example on sorting algorithms, Floyd-Warshall algorithm.

I would recommend extending the research through the questionnaire by more questions and running it across different subjects and levels. In addition to the existing questions, students' perception of themselves could be investigated. They could be asked about their confidence level as mathematicians but also using the computers. Contrasting these responses with their learning from the CAA would allow the investigation whether CAA should be tailored differently for men/women, strong/weak students or keen/not keen computer users. The other aspect for investigation should be students' opinion about the quality of given feedback. They claim that it is the advantage of the CAA, but does it help them to answer the questions correctly next time?

References

Biswal, P.C. (2009) Logic and Set Theory. *Discrete Mathematics and Graph Theory.* 2nd edn. New Delhi: PHI Learning Private Limited, pp. 1-50.

British Dyslexia Association (n.d.) *Eyes and Dyslexia*. [Online]. Available from: http://www.bdadyslexia.org.uk/about-dyslexia/further-information/eyes-and-dyslexia.html#1 [Accessed June 2014].

Brunel University (2008) *MA0422 syllabus* [Online]. Available from: (intranet) (Accessed: February 2010).

Brunel University (2011) *Panel Paperwork 10-11 MA0422.* School of Information Systems, Computing and Mathematics. London: Brunel University.

Bull, J. and McKenna, C. (2004) *Blueprint for computer-assisted assessment.* London: RoutledgeFalmer.

Burton, S. J. et al. (1991) *How to Prepare Better Multiple Choice Test Items: Guidelines for University Faculty*. Brigham Young University Testing Services and The Department of Instructional Science.

Carroll College (n.d.) *MathQUEST/MathVote Resources for Clickers and Classroom Voting in Collegiate Mathematics.* [Online]. Available from: http://mathquest.carroll.edu/ [Accessed March 2015].

Chan, N. and Kennedy, P. E. (2002) *Are multiple-choice exams easier for economics students? A comparison of multiple choice and equivalent constructed response examination questions*. Southern Economic Journal, Vol. 68 (No. 4): 957–971.

Dawkins, P. (2006). *Common Math Errors*. [Online]. http://tutorial.math.lamar.edu/pdf/Common_Math_Errors.pdf (Accessed September 2010).

Design Science (2014) *WebEQ Support.* [Online]. Available from: https://www.dessci.com/en/support/webeq/ [Accessed June 2014].

E-Assessment in Maths Working Group (2013) *Compare systems.* [Online]. Available from: https://mas-hea.ncl.ac.uk/?page_id=221 [Accessed March 2015].

Edexcel (2012) *Specification Edexcel GCSE in Mathematics A.* [Online]. Available from: http://www.edexcel.com/migrationdocuments/GCSE2010/UG029981_GCSE_MathsA_Spec_2012.pdf

Epp, S.S. (2010) *Discrete Mathematics with Applications.* 4th edn. Boston: Brooks/Cole Cengage Learning.

Forehand, M. (2014) *Bloom's Taxonomy.* [Online]. Available from: http://epltt.coe.uga.edu/index.php?title=Bloom%27s_Taxonomy [Accessed June 2014].

Gill, M. (2007) *Development and Evaluation of CAA as a Formative Assessment Tool for University-level Mechanics.* School of Information Systems, Computing and Mathematics: Brunel University. PhD Thesis.

Grassmann, W. K. and Tremblay, J. P. (1996) *Logic and discrete mathematics, a computer science perspective.* New Jersey: Prentice Hall International Editions.

Greenhow, M. and Zaczek, K. (2011) *Embedding computer-aided assessment of mathematics and statistics for first year economics students*. Proceedings of the 10th International Conference for Technology in Mathematics Teaching (ICTMT10) Portsmouth, pp. 122-127.

Greenhow, M., Zaczek, K. and Kamavi A. (2011) *Development and integration of computer-aided assessment of discrete mathematics*. MSOR Connections, Vol. 11 (No. 3): 31-34.

Halpin, J. F. (2006) *Symbolization Guide.* The Logic Café. [Online]. Available from: http://thelogiccafe.net/PLI/symGuide.htm [Accessed March 2014].

Hatch, S. and Hatch, L. (2006) *The GMAT For Dummies.* 5th edn. Hoboken: Wiley Publishing, Inc.

Hebborn, J. (2000) *Decision Mathematics 1.* Oxford: Heinemann Educational Publisher.

Hemmings, S. and Hollows, A. (2005) *Formulating the Research Question.* [Online]. Available from: http://www.socscidiss.bham.ac.uk/research-question.html [Accessed May 2014].

Ikenaga, B. (2009) *Logical Connectives.* [Online]. Available from: http://www.millersville.edu/~bikenaga/math-proof/logical-connectives/logical-connectives.html [Accessed March 2014].

Koshy, T. (2004) *Discrete Mathematics with Applications.* London: Elsevier Academic Press.

Laerd Statistics (2013) *Pearson Product-Moment Correlation.* [Online]. Available from: https://statistics.laerd.com/statistical-guides/pearson-correlation-coefficient-statistical-guide.php [Accessed February 2014].

Leino, K. (n.d.) *Computer Usage and Reading Literacy.* [Online]. Available from: http://www.pisa.no/pdf/Nordisk%20rapport/kap6.pdf

Maplesoft (2014) *Maple T.A.* [Online]. Available from: http://www.maplesoft.com/products/mapleta/ [Accessed August 2014].

Math Goodies (2014) *Introductions to Sets.* [Online]. Available from: http://www.mathgoodies.com/lessons/sets/ [Accessed August 2014].

MathJax (2014) *MathJax.* [Online]. Available from: http://www.mathjax.org/ [Accessed August 2014].

Maths E.G. (n.d.) *Maths E.G.* [Online]. Available from: http://www.mathcentre.ac.uk:8081/mathseg/ [Accessed August 2014].

MDM (2014) *About JavaScript.* [Online]. Available from: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript [Accessed June 2014].

Numbas (2012) *Numbas.* [Online]. Available from: https://numbas.mathcentre.ac.uk/ [Accessed August 2014].

Park, K. (2010) *The development of programming tools for authoring CAA.* School of Information Systems, Computing and Mathematics: Brunel University. Final Year Project.

Pearson (2014) *MyMathLab.* [Online]. Available from: http://www.pearsonmylabandmastering.com/northamerica/mymathlab/ [Accessed August 2014].

Peterson, D. (2013) *Five Steps to Better Tests,* Questionmark White Paper. [Online]. Available from: https://www.questionmark.com/Lists/Resources/Attachments/192/5-Steps-to-Better-Tests-White-Paper-enus.pdf

QTIWorks (2014) *QTIWorks.* [Online]. Available from: https://webapps.ph.ed.ac.uk/qtiworks/ [Accessed August 2014].

Questionmark Perception (2014) *Questionmark Perception.* [Online]. Available from: https://www.questionmark.com/uk/perception/Pages/default.aspx [Accessed May 2014].

Rasila, A. et al. (2010) *Automatic assessment in engineering mathematics: evaluation of the impact*. In Myller, E. (ed.), Reflektori 2010 Tekniikan opetuksen symposium 9.-10.12.2010, 37-45.

Rosen, K.H. (2007) *Discrete Mathematics and its applications.* 6th edn. New York: The McGraw-Hill Companies, Inc.

Sangwin, C. (2013) *Computer Aided Assessment of Mathematics.* 1st edn. Oxford: Oxford University Press.

Scalable Vector Graphics (2014) *Wikipedia*. [Online]. Available from: http://en.wikipedia.org/wiki/Scalable_Vector_Graphics [Accessed August 2014].

Schechter, E. (n.d.) *The most common errors in undergraduate mathematics*. [Online]. http://www.math.vanderbilt.edu/~schectex/commerrs/ (Accessed September 2010)

Shepher, E. and Godwin, J. (2004) *Assessments Through the Learning Process.* https://www.questionmark.com/Lists/Resources/Attachments/101/Assessments_Through_the_Learning_Process.pdf

Shrek (2001) Directed by A. Adamson and V. Jenson [DVD]. USA: DreamWorks Pictures.

Shute, V.J. (2007) *Focus on Formative Feedback.* [Online]. Available from: http://www.ets.org/Media/Research/pdf/RR-07-11.pdf

STACK (n.d.) *STACK - demonstration server*. [Online]. Available from: http://stack.bham.ac.uk/moodle/ [Accessed August 2014].

TechTerms (2014) *Integer*. [Online]. Available from: http://www.techterms.com/definition/integer [Accessed August 2014].

University of the West of England (2014) *The DEWIS e-Assessment System*. [Online]. Available from: http://www.cems.uwe.ac.uk/dewis/welcome/index.html [Accessed August 2014].

Walker, P., Gwynllyw, R. and Henderson, K. (2015) *Diagnosing student errors in e-assessment questions.* Manuscript in preparation.

W3C (2014) *What is MathML?.* [Online]. Available from: http://www.w3.org/Math/ [Accessed June 2014].

W3Schools (2015) JavaScript Array sort() Method. [Online]. Available from: http://www.w3schools.com/jsref/jsref_sort.asp [Accessed March 2015].

WileyPLUS (2014) *WileyPLUS.* [Online]. Available from: https://www.wileyplus.com/WileyCDA/ [Accessed August 2014].

Zaczek, K. (2006) *Computer-Aided Assessment on Logic and Sets.* School of Information Systems, Computing and Mathematics: Brunel University. Final Year Project.

Zaczek, K. and Greenhow, M. (2011) *Assessment of elementary discrete mathematics.* In Green, D. (ed) Proceedings of CETL-MSOR Conference 2010, 6th-7th September 2010, University of Birmingham, pp. 74-80.