ISSN 1112-9867

Special Issue

Available online at http://www.jfas.info

STUDENT OPINION SURVEY ON DELIVERY OF ECE431: COMPUTER PROGRAMMING UNDERGRADUATE COURSE

A. Zabidi[3], I. M. Yassin[1,2,3,*], M. U. Kamaluddin[3], F. H. K. Zaman[3], M. S. A. M.Ali[3], Z. I. Rizman[4] and H. Z. Abidin[3]

[1]Microwave Research Institute (MRI), UniversitiTeknologi MARA, 40450 Shah Alam, Selangor, Malaysia

[2]Malaysia Institute of Transportation (MITRANS), UniversitiTeknologi MARA, 40450 Shah Alam, Selangor, Malaysia

[3]Faculty of Electrical Engineering, UniversitiTeknologi MARA, 40450 Shah Alam, Selangor, Malaysia

[4]Faculty of Electrical Engineering, UniversitiTeknologi MARA, 23000 Dungun, Terengganu, Malaysia

Published online: 10 September 2017

ABSTRACT

Programming is a core subject in many engineering courses. Improvement of programming pedagogy is an active research area with researchers exploring various angles to the issue. This paper presents some findings on Electrical Engineering students' opinions regarding the programming subject that they have taken in their second-year semester. It attempts to discover the programming background experience of the students, their interests towards the subject and their opinions on how the subject is conducted in university. The findings of this paper will be used to help improve the delivery and improve student's understanding of the subject.

Keywords: pedagogy; programming; undergraduate; survey.

Author Correspondence, e-mail: ihsan.yassin@gmail.com

doi: http://dx.doi.org/10.4314/jfas.v9i3s.52

1. INTRODUCTION

Programming is a core subject in many engineering courses. It teaches students how to solve problems and to express them in the form of source code. These skills are essential for students to solve real-world problems when they enter the workforce. A literature study conducted by [1] identified several reasons for this:

1. Programming is highly complex, knowledge-based and practically intensive requiring a high abstraction level.
2. Lack of problem-solving skills by students.
3. Large class sizes limit assistance to students requiring individual attention.
4. Teaching methodologies are primarily based on static contents and do not consider the student's various learning styles.

Improvement of programming pedagogy is an active research area, with researchers exploring various angles to the issue. Generally, programming (especially OOP) has been regarded as a difficult subject by many researchers. This has led to various attempts at improving various aspects of programming pedagogy.

Research by [1] attempted to improve programming pedagogy by giving suggestions to improve the teaching environment in class. Among the recommendations proposed were:

1. Persistent awareness of student's knowledge level.
2. Adapting the teaching method considering the students' learning styles.
3. Use of programming patterns where students are required to complete partial programs instead of writing them from scratch.
4. Including games in study. This would sustain the interest of students while developing their problem-solving abilities.
5. Focus on algorithm design.

In [2], a list of popular electronic books for teaching computer programming was reviewed. It was found that although many e-books include interactive elements to assist student problem visualization, there were some potential areas where e-books could be improved by integrating automatic program visualizations. This would provide the student with a sandbox-type environment, where the students can experiment with various programming concepts to increase their understanding.

In [3], a social code review system called Caesar was proposed to help students learn by peer-review. Caesar is scalable to a large and diverse user population with automated tools for improving review efficiency. Reviews are conducted in a social web interface that stimulatesexchanges between the reviewers. The system was found to significantly improve the code review process. Additionally, its modular structure was found to be suitable as an improved framework for easily extendable code reviewing systems.

Although a highly influential programming paradigm [4-5], a particularly difficult topic to teach in computer programming is Object-Oriented Programming (OOP) [4]-[7]. This may be due to the difficulty of students in grasping the concept of objects and classes in a simplified manner, differentiating them with structured or procedural programming approach [4-5] and to relate and implement them in programming.

Many approaches have been explored by various researchers [9-10]. In [6], a model-driven and object-first approach was explored to teach introductory OOP. The researchers suggested that the teaching of OOP programming should place emphasis on systematic techniques to conceptually model the problem domain in an object-based approach, and emphasis on training the students on applying programming techniques to implement them.

In [5] explains on how to teach OOP at introductory level suggested that OOP can be taught early in the curriculum (with objects being the key focus of the learning process), as students should easily be able to relate between OOP and modelling of real-life objects. Additionally, the paper also criticized the use of C++ as the programming language to teach OOP. This is because C++ can support both procedural and object-oriented programming, which makes it difficult for students to differentiate between them. The paper also suggests the pedagogy should focus on abstraction and design elements in the curriculum.

Research by [4] recommended several changes to how OOP is taught. First, it was suggested that OOP be taught in earlier semesters instead of considering it as an advanced subject introduced late in the curriculum. Second, it should be taught before procedure-based programming. This is because both programming styles are essentially different, and teaching OOP using procedure-based techniques is extremely complicated. Third, the author believes that the tools (Integrated Development Environments (IDEs)) used to teach OOP is unnecessarily difficult and should be made simpler. Additionally, several OOP languages were critically evaluated based on the best criteria for teaching programming language, and it was found that none of the programming languages currently used was suitable for teaching the subject.

In [7], a game-based OOP method was presented to help students to better understand OOP concepts and to help the students to stay interested longer. The proposed Graphical User Interface (GUI)-based method using a software called Greenfoot, which allows interactivevisual objects to manipulate in two-dimensional plane using OOP approaches.

In [8] proposed a constructivism-based approach to teach OOP based on a multi-entity programming problem based on real-life examples. The constructivist approach expects students to complete a series of assignments using Java to model a fast-food chain restaurant well-known by the students. Teaching involved using an object-first approach compared to procedure-based approach with encouraging results.

This paper presents some findings on Electrical Engineering students' opinions regarding the programming subject that they have taken in their second-year semester. It attempts to discover the programming background experience of the students, their interests towards the subject, and their opinions on how the subject is conducted in university. The findings of this paper will be used to help improve the delivery and improve student's understanding of the subject.

2. DESCRIPTION OF ECE431: COMPUTER PROGRAMMING COURSE

The ECE431 course is offered as a first-year core subject for all electrical engineering students enrolled in the EE241: Bachelor of Engineering (Hons) Electronic Engineering and the EE242: Bachelor of Engineering (Hons) Electrical Engineering programs. As an entry-level subject, the syllabus covers the fundamental concepts of programming using the C++ programming language. The course credit hours are three (3 hours lecture and 1-hour laboratory tutorial per week). The student learning time is 120 hours, divided into 56 face-to-face hours and approximately 64 hours of student preparation time (self-study, preparation for tests and assignments, etc.). The objectives of the course are:

1. For the students to be able to write programs with the appropriate syntax.
2. Analyse, design and develop programs to solve selected engineering problems.
3. Present solutions to selected engineering problems.

The subject is evaluated based on two tests, two assignments, a set of quizzes and one final project. The two assignments carry 15% and 30% marks each. Each test contributes to 15% marks each. Meanwhile, quizzes and project carry 15% and 10% each.

Lectures are conducted in two classes per week. Typically, the teaching time for first class is three hours, and one hour for the second tutorial class. In addition to tutorials, tests and quizzes are also conducted during the tutorial. A summary of the chapters and lecture activities are presented in Table 1.

Tutorials are conducted in computer laboratories. CodeBlocks is the software of choice forprogramming due to its free, lightweight and user-friendly nature.

Table 1. Lecture chapters and description

Lecture	Description
Basic C++ Programming	This chapter covers variables types, operators and expressions, standard input and output formatting, program formatting, compilation and execution. The operators covered are arithmetic, relational, bitwise and assignment operator.
Control Flow I	In the first part of this chapter, the students will learn about statements and blocks, as well as the conditional control flow methods (if.., if…else, etc.).
Control Flow II	In the second part of this chapter, focus is given to repetitive control structures such as while, do..while, for, break and continue.
User Defined Functions	This chapter covers the programming of user-defined functions (both void and return value types), passing by argument and passing by pointer, function prototypes, global and local variables, header files and block structures.
Arrays I	The chapter is also divided into two. This first part concentrates on basic one-dimensional arrays, array processing and passing of arrays to functions.
Arrays II	Extends the array processing concept with multi-dimensional arrays.
Pointers and Arrays	Covers pointers basics, addressing, pointers as function arguments, relationship between pointers and arrays. Address arithmetics, character

	arrays and pointer arrays are also covered.
File Processing	This chapter covers file processing commands to write/read data to/from files. Only text-based sequential file access is covered.
Structures	Students will be introduced to the basics of structures and its applications. They will also learn about structure operators (dot and arrow), passing structures to functions, structure enumeration and structure union.
Object Oriented Programming (OOP)	The basic concepts of OOP are introduced in this chapter. Students learn about classes and objects, constructors and destructors, as well as inheritance and polymorphism.

3. QUESTIONNAIRE DESIGN

The questionnaire was design to assess three important aspects of study namely the student's background and interests, their opinions on the assessment and delivery methods of the course, and self-assessment of their skills. The Malay language was used in the questionnaireas we believe that it would facilitate understanding of the questions as it is the primary language spoken by the students in their daily communication. The original questions and their English translations are shown in Table 2.

Table 2. List of questions in questionnaire and its English

No	Question (Malay)	English Translation
1.	Jantinaanda?	What is your gender?
2.	Berdasarkanskala 1-5, bagaimanaandatakrifkankebolehanpengaturcaraananda?	Between a scale of 1-5, how do you rate your current programming

		ability?
3.	Adakahandapernahmengambilsubjekpengaturcaraan di peringkatsekolah?	Have you taken programming at school level?
4.	Adakahandapernahmengambilsubjekpengaturcaraan di peringkat diploma?	Have you taken programming at diploma level?
5.	Adakahandaberminatdengansubjekpengaturcaraan?	Are you interested in programming
6.	Merujukkepadasoalan 5, silajelaskankenapaandaminat/tidakberminatdengansubjekpengaturcaraan.	Based on question A-5, why are you interested/not interested in programming
7.	Berdasarkanpendapatanda, berapapentingnyakemahiranpengaturcaraansepanjangpengajianandadalambidangKejuruteraanElektrik?	In your opinion, how important do you think programming is throughout your study in Electrical Engineering?

8.	Berdasarkanpendapatanda, berapapentingnyakemahiranpengaturcaraandalammasahadapankerjayasebagaiJuruteraElektrik?	In your opinion, how important do you think programming is in your future career as an electrica engineer?
9.	Berdasarkanciri-ciriberikut, apakah yang andarasakanfaktorpenting yang perluadapadaseseorangpensyarah yang mengajarsubjekini?	Based on these characteristics what do you think are important factors that lecturers need to possess to teach this subject?
10.	Berdasarkanskala 1-5, berapakahkesesuaiansubjekpengaturcaraandisampaikandalamkuliahdewan (mass lecture)?	Based on a scale of 1-5, how suitable is programming to be delivere in a mass lecture?
11.	Apakah yang bolehditambahbaikdarisegicarapengajaranpensyarahuntukmembantupembelajarananda?	What can be

		improved in terms of course delivery in order to improve your learning experience?
12.	Berdasarkanskala 1-5, apakahnisbah yang sesuaiantarateoridanpraktikalbagisubjekpengaturcaraan?	Based on a scale of 1-5, what is a suitable ratio between theory and practical for a programming subject?
13.	Berdasarkanskala 1-5, bagaimanakahandalihatnisbahmasa yang patutdiperuntukkanantarakuliahdan tutorial bagisubjekpengaturcaraan?	Based on a scale of 1-5, what is a suitable time ratio that should be allocated between lecture and tutorial for this subject?
14.	Sejauhmanacarapenilaian di bawahberkesanuntukmembantupemahamanandadalamsubjekini?	How effective

		are these assessment methods to help your understanding of the subject?
15.	Bagaimanaandalihattahapkesesuaianmakmaluntukmenjalankan tutorial anda?	Is the provided laboratory suitable for your tutorial learning experience?
16.	BagaimanaandalihattahapkesesuaianperisianCodeBlocksuntukmenjalankan tutorial anda?	How suitable is the CodeBlocks software to do your tutorial exercises?
17.	Apakah yang bolehditambahbaikdarisegiprasaranamakmaluntukmembantupembelajarananda?	What can be improved in terms of laboratory facilities to help improve your learning experience?
18.	Silanilaikantahapkefahamanandabagisetiaptopikberikut	Please rate your

		understanding for each of these topics.
19.	Sejauhmanapemahamanandadalamteknikrekabentuk program berikut?	How much do you understand these programming design techniques?
20.	Sejauhmanapengetahuanberkenaanteknikrekabentuk program pentingdalamsubjekpengaturcaraan?	How important are programming design to programming?
21.	Berapaperatusmasa yang andaperuntukkanuntukmengulangkajisubjekpengaturcaraanberbandingdengansubjek lain yang andaambil semester ini?	How many percentage of your home study time was used to study this subject relative to others subjects that you are taking this semester?

4. RESULTS AND DISCUSSION

A total of 44 respondents participated in the survey. The respondents were taking the course for the September-January 2016 semester. The students had recently graduated from their Diploma in Electrical Engineering program with various specializations. Therefore, their programming skills were varied as different specializations had different focus during theComputer Engineering program. The results of the questionnaire are discussed in Section 4-A to Section 4-C.

4.1. General Information and Programming Background

In this section, respondents were asked basic questions about their background and interest in programming. The gender breakdown of the respondents is shown in Fig. 1. Overall, 27 of the respondents were male while the remainder was female.

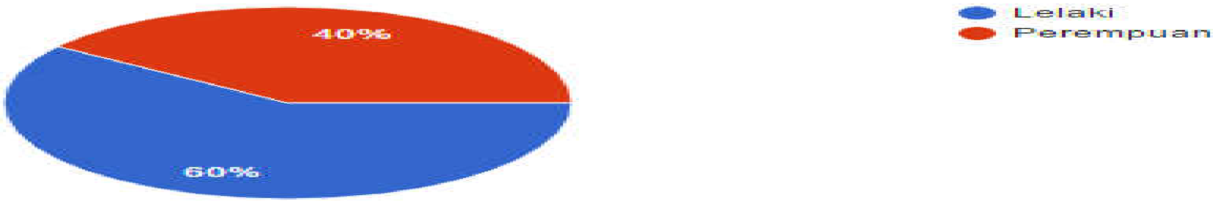


Fig.1. Gender distribution breakdown of respondents (male: lelaki, female: perempuan)

The respondents were asked to self-assess their programming skills from a scale of 1 (very poor) to 5 (excellent) (Fig. 2). It was found that most respondents ranked their programming skills as either average, very high or excellent. This indicates that the respondents could understand the curriculum well.

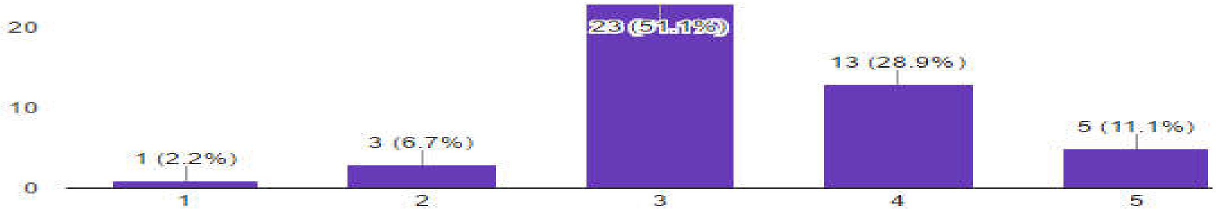


Fig.2. Self-assessment of programming skills (1: very poor to 5: excellent)

Additional questions A-3 and A-4 revealed that most respondents had formal exposure to programming at diploma level (Fig. 3 and Fig. 4). Only a small percentage of respondents were formally introduced to programming at primary or secondary school level. In recognition of Information Technology in the development of the country, the Malaysian government has introduced a new subject in Information Technology (IT) which several of the respondents enrolled. Additionally, invention competitions such as the Malaysian Technology Expo, International Invention and Innovation Exhibition may have also been a factor for the relatively early exposure at secondary education level.

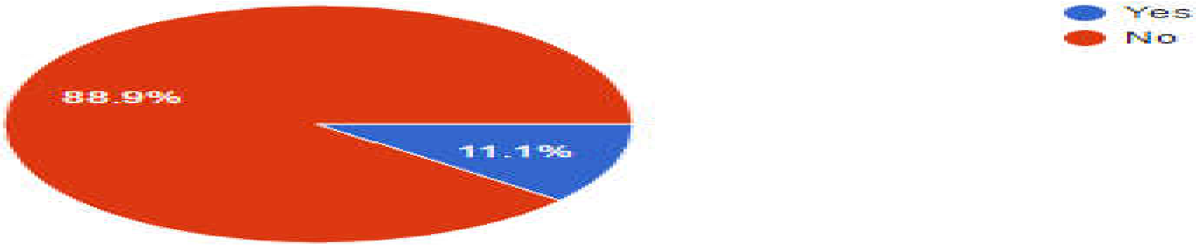


Fig.3. Exposure to programming at secondary school level

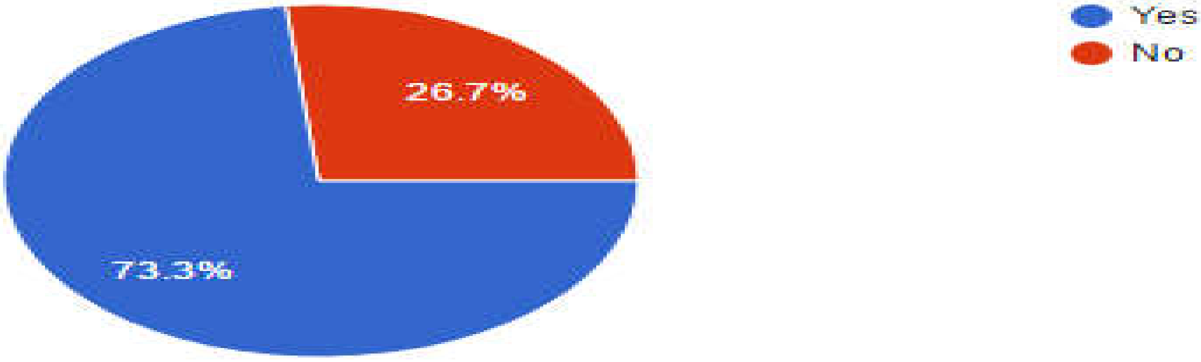


Fig.4. Exposure to programming at diploma level

In question A-5, the respondents were asked to rate their interest in programming (Fig. 5). A strong majority of the students expressed high interest in programming. When asked to explain for the interest, the students expressed the reasons as shown in Table 2. Summarizing the responses, the respondents understand that programming has many potential applications, opens opportunities in the engineering, and its ability to enhance their thinking skills. Many respondents expressed desire to learn something new and satisfaction in solving problems as the primary reason to enjoy programming.

Non-interested respondents attribute their lack of interest to the difficulty to master the concept, logic and syntax involved. This may lead to the respondents being frustrated with the subject as more effort is required to understand the subject. One respondent remarked that programming is not a mechanical (movement-oriented) subject, which may suggest that this student is a spatial/kinesthetic type learner.

When asked to rate the importance of programming to their currently enrolled program (Fig. 6) and their future careers as electrical engineers (Fig. 7), a significant majority agrees on the importance of programming for both their study and future careers.

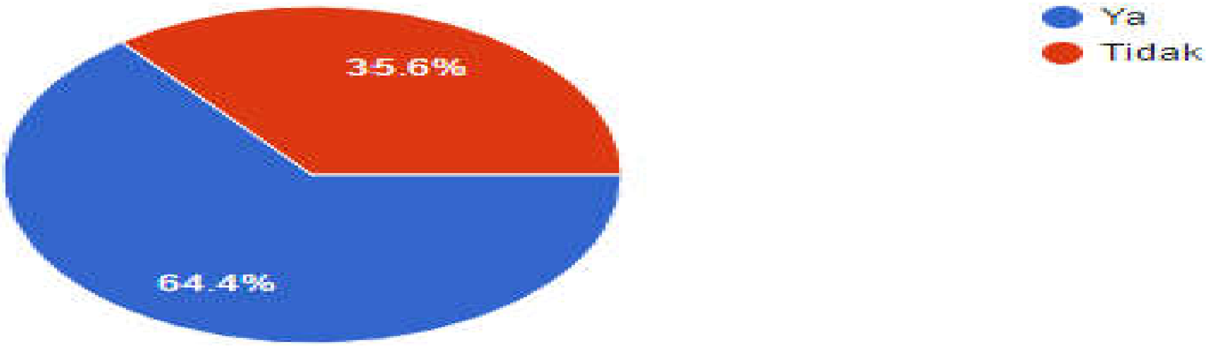


Fig.5. Interest in programming (ya: yes, tidak: no)

Table 3. Reasons for interest/non-interest in computer programming

Reason	Number of Comments
Reasons for Interest in Computer Programming	
Satisfaction of successfully solving a problem/creating something new	6
Realization of its many potential applications	3
Development of soft skills	1
Understanding of how technology works	1
Appeals to my organized nature	1
Potential for income generation/jobs in the future	3
It is easy for me to understand	2
I enjoy learning something new	10
It can prepare me for my future subjects	1
I enjoy a challenge	1
It develops my ability to think creatively/systematically	3
Reasons for Non-Interest in Computer Programming	
Programming is complicated/hard to understand	8
Programming is a taxing task	1
I am not interested in programming	2
I do not understand the syntax	1
I do not get the concept	2
It does not involve something mechanical	1

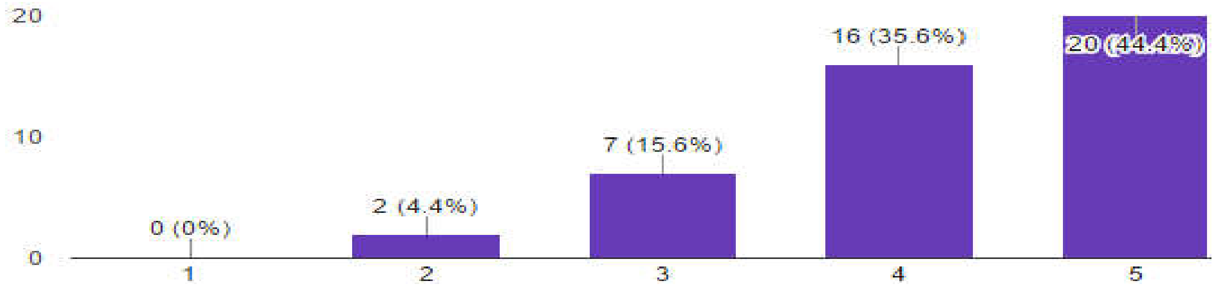


Fig.6. Response when respondents were asked to rate the importance of programming to currently enrolled program (1: not important, 5: very important)

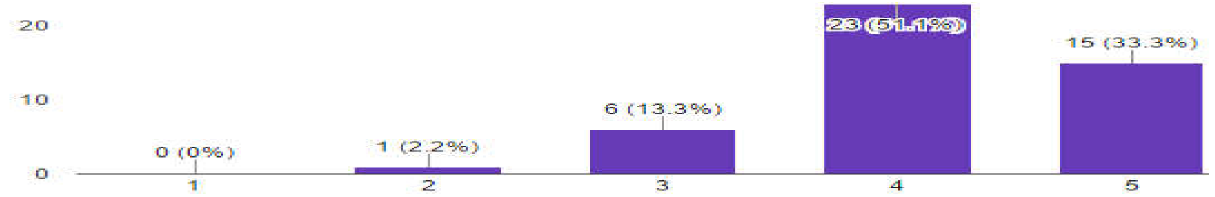
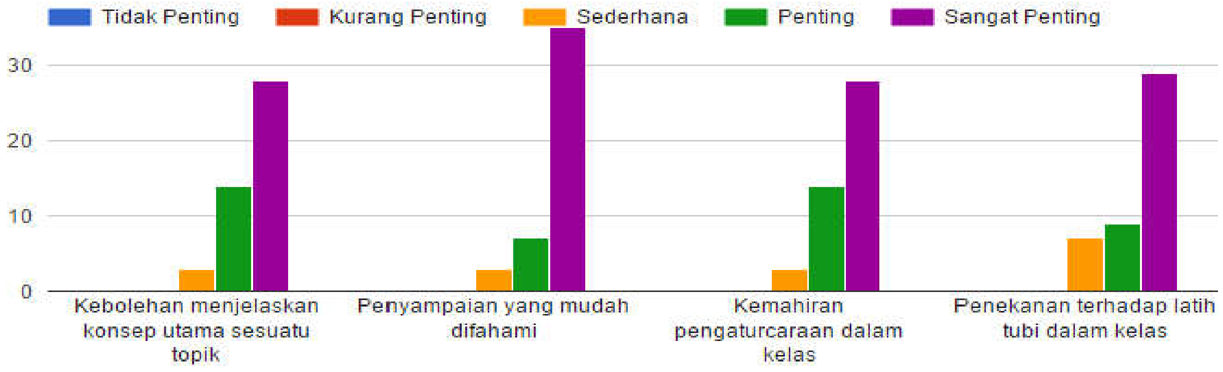


Fig.7. Response when respondents were asked to rate the importance of programming to their future careers as electrical engineers (1: not important, 5: very important)

4.2. Delivery and Assessment Methods

The second part of the questionnaire is focused on the respondent's opinion on the delivery and assessment method of the subject.

In the first question, the respondents were asked to rate the characteristics that they find favourable for a lecturer to teach the subject (Fig. 8). The respondents placed a heavy emphasis on concepts, understandable delivery and practice as compared to theoretical aspects, as shown by relatively low rating for theoretical delivery (Graph 5) compared to the other questions. Based on this, lecturers may want to adjust their teaching methods to focus more on practical aspects and reduce delivery of theory in class. Another possible improvement is increasing the time allocation for tutorials relative to lectures in class. Currently, the course is divided into three hours teaching and one hour tutorial per week.



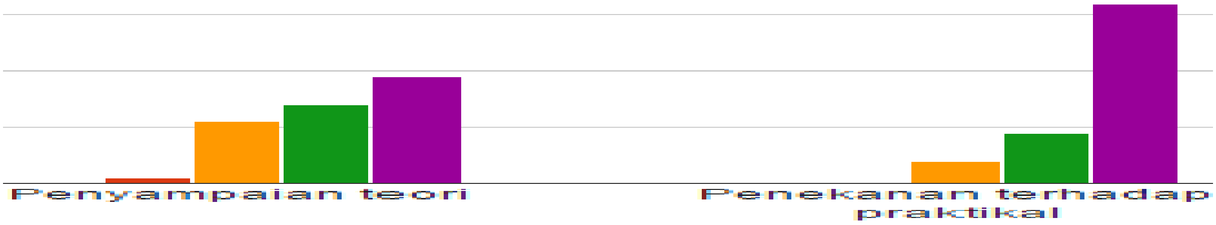


Fig.8. Respondents were asked to rate the skills necessary for a lecturer to teach the subject (blue: not important, red: less important, orange: average, green: important, purple: very important)

In the second question, students were asked to rate the suitability of the course to be conducted as mass lecture in the lecture hall (Fig. 9). The results indicate that the respondents were largely satisfied with the course being conducted in mass lecture mode.

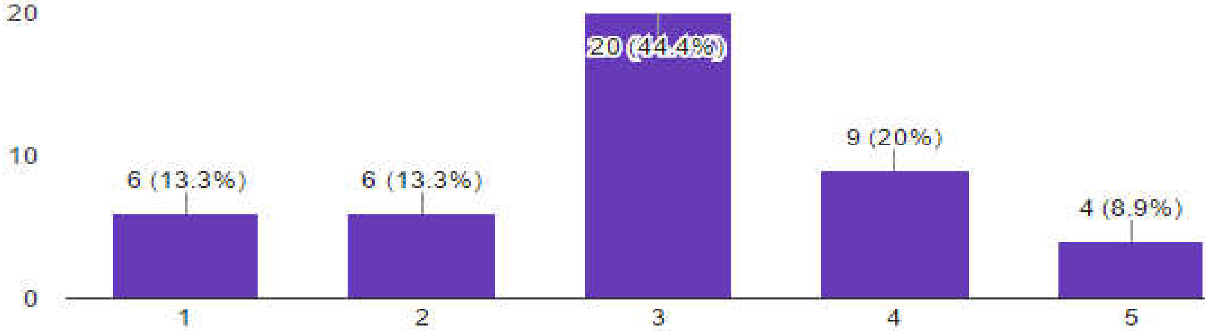


Fig.9. Response when respondents were asked to rate the effectiveness of the course to be conducted as mass lecture (1: not suitable, 5: very suitable)

Recommendations were sought from the respondents on how to improve the delivery of the course (Table 3). Most of the responses were focused on these areas:

1. Lectures and tutorials need to be conducted in smaller groups (items 1.1, 2.2 and 2.3)
2. Lecturer teaching experience, conduct in class and his/her ability to convey and visualize concepts is considered particularly important (items 1.2, 1.4, 3.2). The students also prefer the same lecturer for both lecture and tutorial to avoid confusion, and do not prefer research assistants with relatively little experience compared to lecturers.
3. Focus towards practical-based delivery compared to theory with more examples (items 1.5, 2.1 and 3.1). This is further confirmed in Fig. 10 and Fig. 11, where the students were asked about the ideal focus and time allocation for theory versus practical.

Table 4. Respondent suggestions on how to improve subject delivery

No	Suggestions	Number of

		Comments
1.0	**Lecture Delivery**	
1.1	Lectures need to be conducted in the laboratory/in smaller groups	5
1.2	Lecturers need to improve their presentation skills/delivery of materials/visualize materials for ease of understanding	2
1.3	Lectures should simplify the delivery of materials/give emphasis on concepts	1
1.4	Lecturers need to have a good personality	3
1.5	I need more examples in class	3
1.6	Lecturers should give more time for students to absorb the knowledge	1
1.7	Lecturers should rely less on Research Assistants (RAs) to conduct classes as they are less effective than lecturers	1
2.0	**Tutorial Delivery**	
2.1	Programs should be more focused to real-life problem-solving rather than "fill in the blanks"	2
2.2	I need one-to-one coaching	1
2.3	Give tutorials in smaller groups	1
3.0	**General**	
3.1	More emphasis/time needs to be given on practical compared to theory	7
3.2	Do not use different lecturers for tutorial and lectures	2

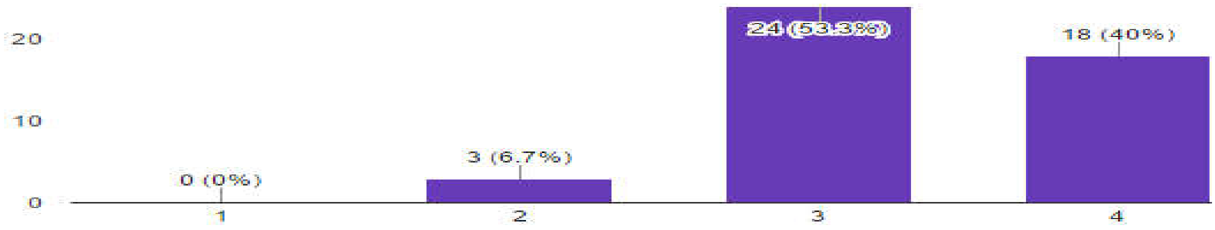


Fig.10. Response when respondents were about the ideal focus for theory vs. practical (1: fully theory-based, 4: fully practical-based)

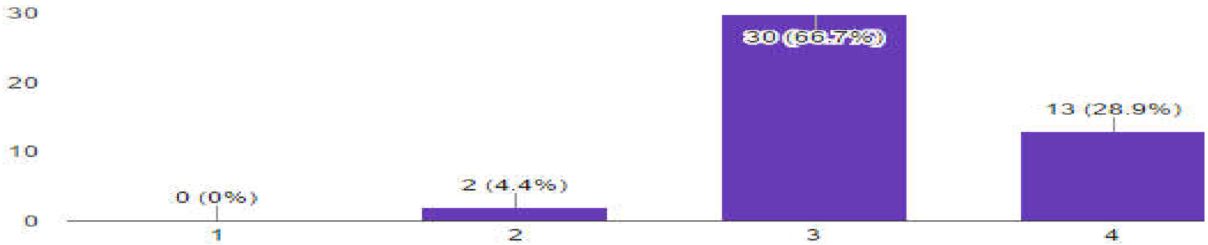


Fig.11. Response when respondents were about the ideal time allocation for theory vs. practical (1: maximum time for theory, 4: maximum time for practical)

When asked about the suitability of assessment methods (Fig. 12), the students agreed that the current assessment method was adequate. However, the students less preferred quizzes, possibly because of the high frequency and “fill in the blanks” nature of the quizzes.

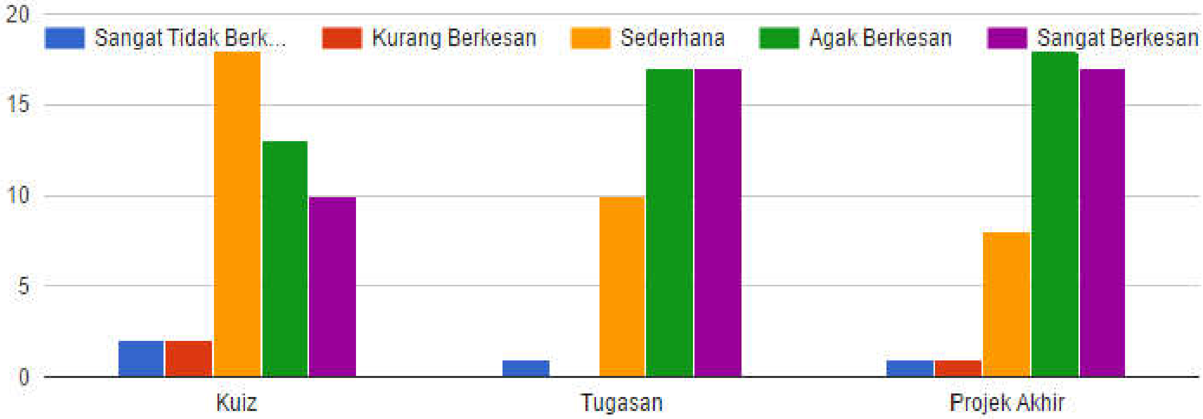


Fig.12. Suitability of assessment methods for the course (blue: very ineffective, purple: very effective, kuiz: quiz, tugasan: assignment, projek akhir: final project)

The students were then asked to rate whether the laboratory and CodeBlocks Integrated Development Environment (IDE) software used were sufficiently conducive for their learning experience (Fig. 13 and Fig. 14). It was found that the students were generally happy with the facilities and software used with minor comments on some problematic computers and projectors.

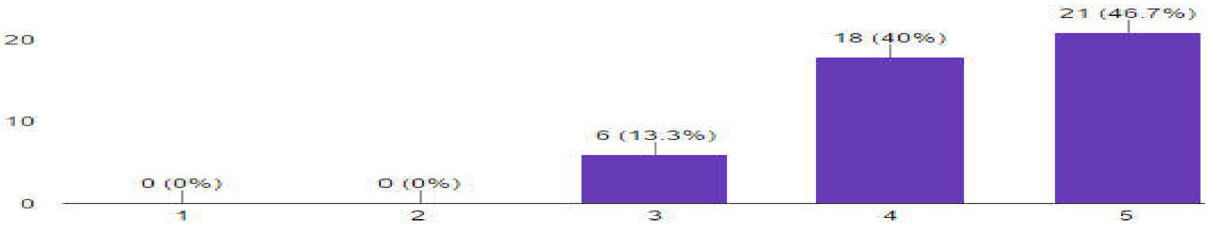


Fig.13. Responses when students were asked whether the laboratories and facilities provided were sufficiently conducive to their learning experience (1: very inconducive, 4: very conducive)

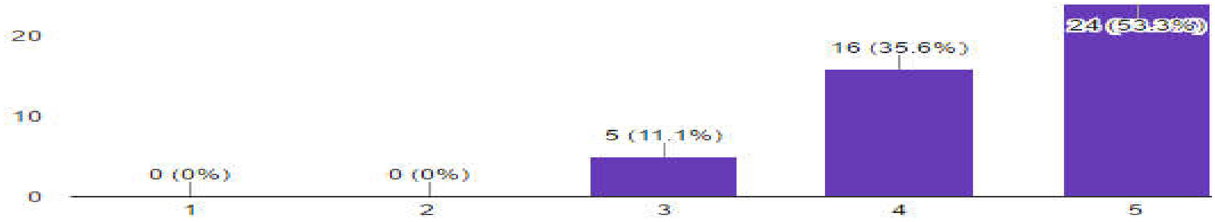


Fig.14. Responses when students were asked whether the CodeBlocks software used were sufficiently conducive to their learning experience (1: very inconducive, 4: very conducive).

4.3. Personal Assessment

In this section, the students are requested to self-assess their understanding of subject's chapters to determine potential delivery areas that can be improved. Most students showed average understanding for each chapter, and it appears that there was a decrease of understanding in the more complex chapters. From the results, there appears that there is potential for improvement in two chapters namely Structures and File Operations. These chapters may benefit from more exercises and examples to improve student's understanding.

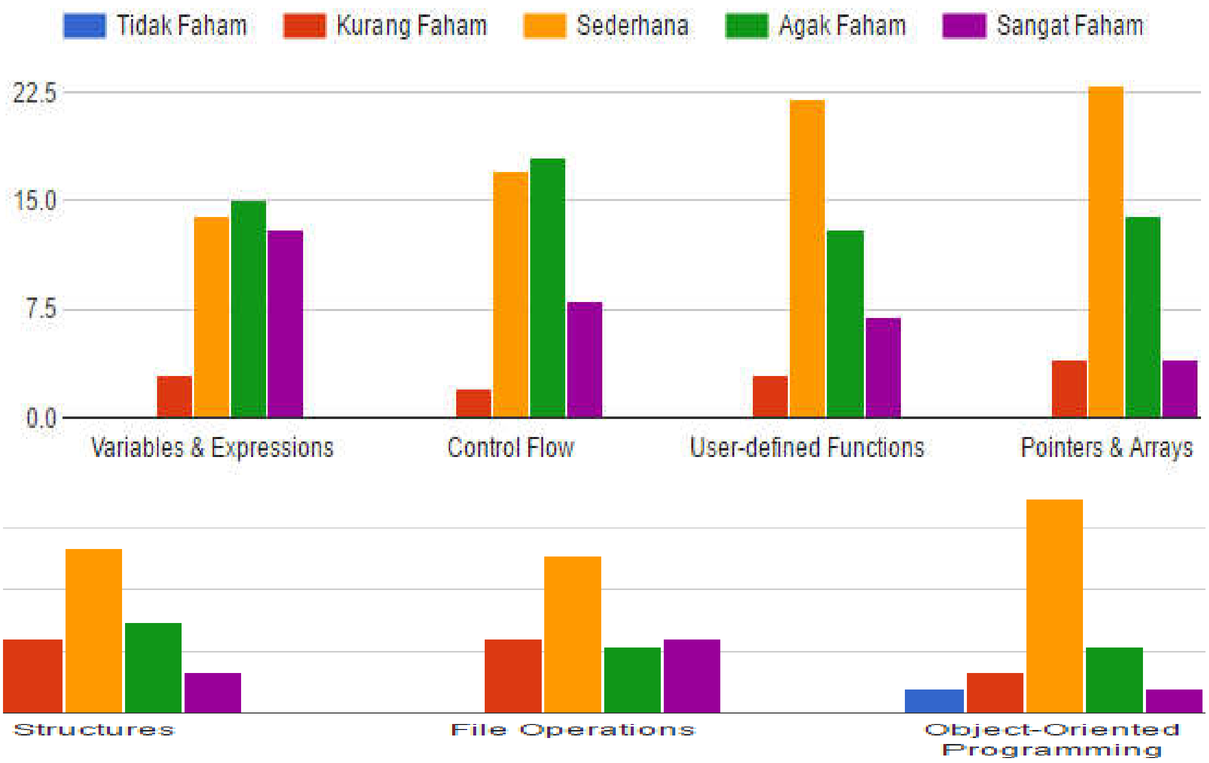


Fig.15. Course understanding by chapter (blue: no understanding, purple: very high understanding)

Since program design is also an important aspect of programming design, we asked the students whether they are adept at several common programming methods (Fig. 16).It appears that the students were generally able to understand and implement the programming design methods. Of all the design methods, it appears that flowchart is the design method that the students are most comfortable with possibly because it was popularly used during their diploma-level studies. The students also showed a deep appreciation of the programming design methods and how they can improve their programming.

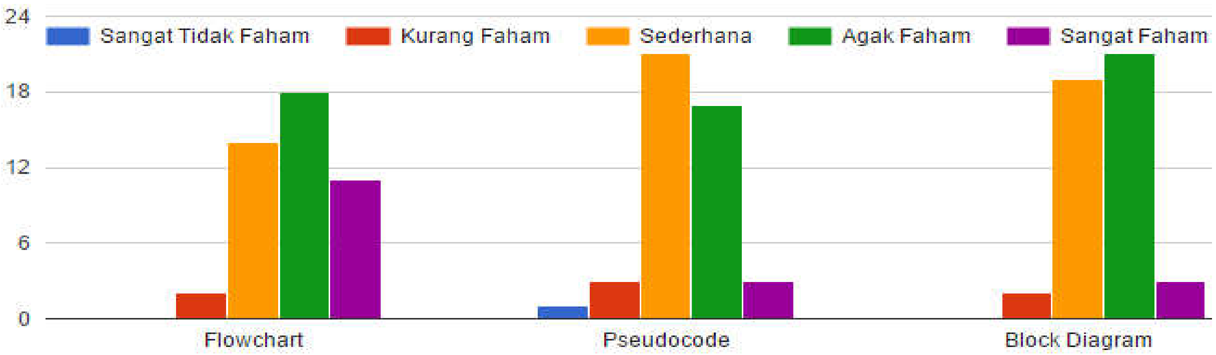


Fig.16. Understanding of program design method (blue: no understanding, purple: very high understanding)

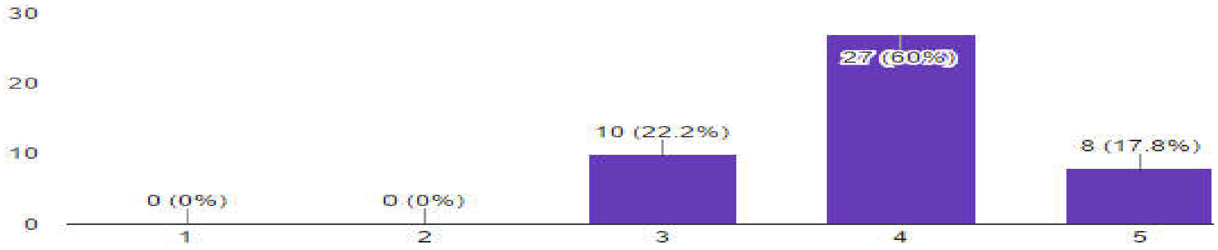


Fig.17. Respondent's opinion on the importance of programming design methods (1: not important, 5: very important)

When asked about how the percentage of time they spent studying at home for the subject relative to others, the results show that in average the students spent between 60 to 80% of study time for the subject. This is a high percentage indicative of the students' opinion on the difficulty of the subject.

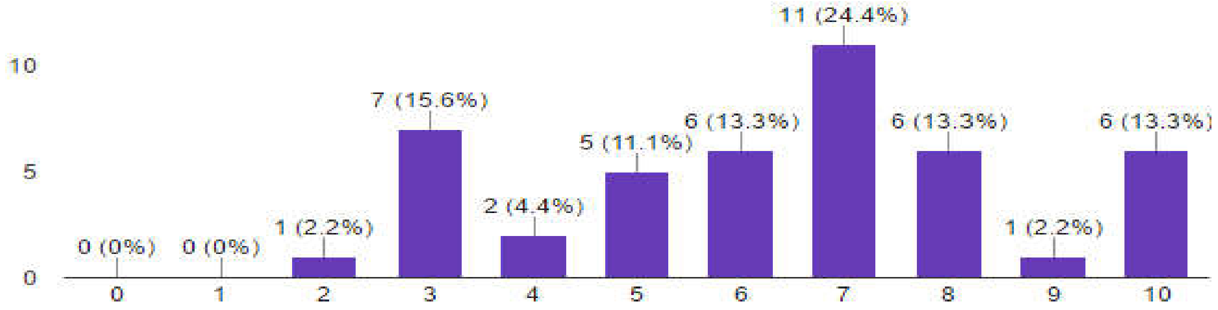


Fig.18. Respondent's percent of time spent studying the subject (scale from 0% to 100%)

5. CONCLUSION

A study was conducted to gather information on students' opinions on their ECE431: Computer Programming learning experience. Three categories of questions were asked namely 1) Programming background, 2) Delivery and assessment methods, and 3) Self-assessment of programming skills. In the first assessment, it was found that the students were primarily exposed to programming at tertiary education level were generally interested in programming and realized the importance of programming for their current program and career.

In the second category, we discovered that the students preferred the delivery to be practical-based with focus on concepts, exercises and examples in smaller classes compared to theory and mass lectures. The assessment methods and laboratory facilities were generally considered to be satisfactory.

In the third category, the students were found to be adept at the flowchart program design method. However, they appear to have difficulty in grasping the concept in two chapters which is an opportunity for the lecturers to improve.

Although the results were limited on undergraduates taking the course at degree entry level in UniversitiTeknologi MARA, we hope that our findings would be able to help higher learning institutions in Malaysia particularly and the world generally to design the curriculum better to fit the students' interest and needs.

6. ACKNOWLEDGEMENTS

The authors would like to thank UniversitiTeknologi MARA (UiTM) Malaysia for the funding for this research (grant number: 600-RMI/DANA 5/3/ARAS (70/2015) and FRGS/1/2015/ICT03/UiTM/02/4).

7. REFERENCES

[1] Gomes A, Mendes A J. An environment to improve programming education. InACM International Conference on Computer Systems and Technologies, 2007, pp. 1-6

[2] Unanue R M, Velasco M P,Flores C P,Fuentes J U, Iturbide J Á V. Electronic books for programming education: A review and future prospects. ACM SIGCSE Bulletin, 2002, 34(3):34-38

[3] Tang M. Caesar: A social code review tool for programming education. Master thesis, Cambridge: Massachusetts Institute of Technology, 2011

[4] Kölling M. The problem of teaching object-oriented programming: Part 1. Journal of Object Oriented Programming, 1999, 11(8):8-15

[5] Okur M C. Teaching object oriented programming at the introductory level. Journal of Yasar University, 2007, 1(2):149-157

[6] Viswanathan K V. Teaching object-oriented programming. Journal of Object-Oriented Programming,1996, 9(2):1-4

[7] Yan L. Teaching object-oriented programming with games. In6th IEEE International Conference on Information Technology: New Generations, 2009, pp. 969-974

[8] Thramboulidis K C. A sequence of assignments to teach object-oriented programming: A constructivism design-first approach. Informatics in Education-An International Journal, 2003, 2(1):103-122

[9] FauziA, Rizman Z I. Field trip education approach beyond classroom: Microwave course case.Mediterranean Journal of Social Sciences, 2015,6(4 S1):89-94

[10] Latip M F A, Udin M K A M, Othman M M, Yassin I M, Rizman Z I,Zaini N, Hidayat M N, Aminuddin N, Herman S H, Saad H, Rahiman M H F, Adnan R. Implementation of fuzzy logic-based final year project student-supervisor matching system.International Journal of Advanced and Applied Sciences, 2017,4(4):159-163

How to cite this article:
Zabidi A, Yassin M I, Kamaluddin M U, Zaman F H K, Ali M S A M, Rizman Z I, Abidin H Z. Student opinion survey on delivery of ece431: computer programming undergraduate course. J. Fundam. Appl. Sci., 2017, *9(3S), 662-684*