ADAPTIVE ASSESSMENT

IN THE CLASS OF PROGRAMMING

Dimitra I. Chatzopoulou A.M. 06/18

Supervisor Instructor: Anastasios A. Economides

Information Systems Department, University of Macedonia.

2009

<u>**INDEX**</u>

ADAPTIVE ASSESSMENT IN THE CLASS OF PROGRAMMING

Dimitra I. Chatzopoulou and Anastasios A. Economides
Information Systems Department
University of Macedonia, Greece

Abstract

This paper presents P.A.T. (Programming Adaptive Testing), a computerized adaptive testing system for assessing students' programming knowledge. P.A.T. was used in two high school programming classes by 73 students. After research was carried out, it was found helpful in increasing students' cognitive domain skills. In addition, it assists them to discover their shortcomings in the teaching material. P.A.T. helps teachers to assess their pupils with objectivity. Finally, P.A.T. classifies students according to their programming skills in three Levels of knowledge and research results showed that it successfully predicts students' performance in the National Exams.

Keywords: computerized adaptive testing, adaptive assessment, programming testing, programming assessment, programming skills.

1. Introduction

Programming comprises a broad scientific field that demands not just immaculate theoretical knowledge, but also deep understanding of the framework of Structured Programming. Moreover, students need to have a deep understanding of the syntax of the language they are called upon to learn, in order to practice. People involved in Programming realize that the Science of Programming requires perfect handling of the Logic behind the idea, rather than ability of memorizing the syntax of different languages.

It is not uncommon that several students, upon completing a year of study on Programming, exhibit serious shortcomings on basic Programming knowledge (McCracken et al., 2001). It was found that students with little or no practical work were able to produce a piece of code in the final traditional way of assessment through memorization and achieve a "good" grade in the course (Woit & Mason, 2003). Furthermore, it is difficult to closely observe the progress of a particular student, especially in large classes. This happens because there is not enough available time for the teacher to interact personally with every student. Teaching and learning Programming has created significant difficulties to both teachers and students (Wang & Wong, 2008). Innovative ways are needed in order to improve the effectiveness of teaching Programming. Assessing the students' programming knowledge using computers in a regular and continuous basis could help. The assessment results could be used for continuous improvement of teaching effectiveness and learning quality (Khamis, Indris, Ahmad and Idris, 2008).

The assessment should be carefully designed according to pedagogical theories. Lister and Leaney (2003a) encouraged teachers to design assignments according to the cognitive levels defined in the Taxonomy of Educational Objectives (Bloom, 1956). These levels are the following (from lowest to highest): 1) Recall of data, 2)

Comprehension, 3) Application, 4) Analysis, 5) Synthesis, and 6) Evaluation. However, it is difficult to categorize a question into the proper cognitive level (Thomson at al., 2008). Bloom's Taxonomy can be also used in the course design (Scott, 2003). Oliver and Dobele (2007) argued that the lower cognitive levels (Recall of data, Comprehension, and Application) should be gained during the first year of studies. Subsequently, the students could become able to move onto assessments that require higher cognitive levels (Analysis, Synthesis and Evaluation). Otherwise the assessment will have a negative effect on students to make "upward progress".

This study presents P.A.T., a computerized adaptive testing system for assessing students' programming skills. The questions are categorized both into three difficulty levels and into three cognitive levels (Recall of data, Comprehension, and Application). If a student answers correctly a question, the next question is more difficult. Otherwise, the next question is easier.

The next section 2, presents types of computerized assessment. Section 3 presents P.A.T., a multiple choice questions testing system, which was developed and used in a high school programming class for novice programmers. In section 4 the questions content is presented and how we classify them. Research results are performed in section 5. The most significant section is 6 where P.A.T. achievement is presented in which students are classified according to their programming skills in order to predict their performance in National Exams. Finally in section 7 the strengths and weakness of the model are illustrated.

2. Computerized Testing of Programming Skills

Computerized assessment offers speed, availability, consistency and objectivity of the assessment (Ala-Mutka, 2005). In order to assess programming skills, two types of computerized assessment could be used: 1) Code Writing, and 2) Multiple Choice Questions (MCQs).

One of the problems faced by Computer Science instructors is bridging the following two gaps: 1) gap between the course and what to teach, and 2) gap between what the students had been taught and how to assess this knowledge (Starr, Manaris and Stavley, 2008). This means that even if two schools offer the same course in Computer Science, the assessment can be different from one school to other because the teachers' objectives and teaching as well the students' demands may vary.

Whalley et al. (2006) showed that novice programmers were not yet able to work at fully "abstract level" (high cognitive level). So, students that can not read a short piece of code and describe it are not capable intellectually to write code by themselves. Thus, it is better to assess novice programmers using MCQs. On the other hand, if the students are at an advanced level and the course focus is on developing the students' programming skills then it is better to use Code Writing Assessment. Of course, a combination of both types of assessment could be also used.

Next, both types of computerized testing of the students' programming skills are presented.

2.1. Computerized Testing using Code Writing exercises

There are many ways to solve a problem in a programming language and more specifically in high level programming languages. So, many instructors prefer to correct manually the “solutions” given by the students. Ala-Mutka (2005) found that 74% of instructors preferred the “practical work of assessment”. However, the manual inspection of the code is inefficient and the possibility to over or under estimate a student is increased (Kolb, 1984) depending on the number of students.

Computerized testing could help in achieving accurate estimation of the student’s knowledge. However, the design of a Code Correction and Assessment system presents many difficulties regarding its objectivity (Schwieren et al., 2006). Furthermore, both instructors and students should become familiar with such a system.

Code Writing Assessment systems could be divided into fully automatic and semi-automatic systems (Ahoniemy et al., 2008). The semi-automatic systems are used if the students are novice in this experience. In this case, they need a feedback from a human. Also, the quality and efficiency of the source code are very hard or unfeasible to be evaluated via a fully automated system. Next, the following semi-automatic tools are presented: ALOHA, Sakai, EMMA, ASSYST and TRY.

ALOHA (Ahoniemy et al., 2008) bases its objectivity[1] on the use of “rubrics” (Becker, 2003). ALOHA provides to the grader the work list (list of students’ submissions). The teacher is responsible for adding grades into the rubric that defines not only the grading process but also some template feedback phrases. Finally, the ALOHA’s objectivity is examined using statistical analysis of the grading distribution of the graders using Aloha (Ahoniemi and Reinikainen, 2006).

Sakai is used in a Java course assessment (Suleman, 2008). Assignments can be compiled, tested, executed and scored without human intervention. This means that the workload is reduced and all the submissions are marked with exactly the same criteria. Also, the feedback helps the students to learn from their mistakes. The submitted assignments are sent to the Automatic Marker which compares the output to a predefined one. If the solution is not correct then the student is given a list of the produced output and the expected one.

EMMA is a web-based tool and is used in a Java course assessment (Tanaka-Ishii et al., 2004). Students’ programs are executed and tested on different inputs. Then they are examined and graded by several teachers and graduate students. Furthermore, “notable results” can be seen by all the students.

In ASSYST (Jackson & Usher (1997), the students submit their assignments via e-mail. Instructors run the system which tests and marks the submitted programs. Then the students receive an evaluation report. The final mark is based on the quality of the source code, the efficiency (e.g. lines of code) and the effectiveness of the submitted program.

[1] According to Habeshaw et al. (1992) the objectivity can be achieved only through Multiple Choice Questions.

In TRY (Reek, 1989), the students submit their programs and the instructors test them. The output evaluation is based on textual comparison.

Next, the following fully-automatic tools are presented: PASS3, Oto Marmoset, and BOSS.

PASS3 provides both immediate feedback to the student regarding his/her submission and a detailed performance statistic regarding his/her progress (Choy et al., 2008). The difference with the previous version of PASS (Yu et al., 2006) is that there multiple levels of difficulty regarding the programming activities and a student selects the level according to his/her capabilities (Wang & Wong, 2008). In addition, the system can help the student by presenting to him pre-stored hints. It offers a learning environment in contrast to others which support an environment for assignments (Tanaka-Ishii et al., 2004).

Oto is a marking tool that provides support for submission and marking of assignments in a programming course (Tremblay et al., 2007). First, it tests the student's submission (i.e. program) to ensure that it exhibits correct behavior. Then it evaluates its structure and style to ensure that the appropriate standards have been followed. Finally, it sends the grade and the marking report to the student.

Marmoset monitors the student's progress and sends a feedback to both the student and the instructor (Spacco et al., 2006). It can be used in several programming courses (e.g. C, Java). The assignments are sent by the Submit Server and tested by the Build Server.

BOSS (Joy et al., 2005) supports both submission and testing of programs in various programming languages (e.g. Java). It compares the student's submission to the correct one.

The aforementioned tools helped to the creation of the xlx System (Schwieren et al., 2006). The code of this system can be evaluated through Static and Dynamic control. The Static control checks the source code for syntactic errors. The Dynamic control additionally examines the code's performance, structure and output produced after its execution, in relation to a standard code.

In a semi-automatic system, the grading is performed by a human. In a fully-automatic system, a student's program that does not meet the assessment's criteria can not get partial marks (Suleman, 2008). Also, the assignment can be examined with respect to the correctness of the output but not to the styling (readability) and efficiency[2]. The common disadvantages of both semi-automatic and fully-automatic tools are that the complexity of the level of programming must be simple enough in order to be measurable and that they can not examine students' programs at an abstract level (e.g. meaningfulness of variables). Furthermore, the student must follow strict steps in order to complete his/her assessment.

[2] According Hwang et al. (2008) the efficiency is testified if program runs correctly. The elements that measure efficiency are "algorithms and data structure".

2.2. Computerized Testing using Multiple Choice Questions

It is a common belief among many (Traynor & Gibson, 2005) in the field of education that multiple choice questions tests are the easy and the lazy way to assess students. However, research (Lister & Leaney, 2003b) has proved that quality multiple choice questions is by no means “the work of the lazy”.
According to Lister (2005), Assessment through Multiple Choice Questions can be effectively administered to beginner programmers, who have acquired basic skills. If a student scores poorly or averagely on basic skills, s/he is bound to fail on final exams, which are comparatively more demanding and require more knowledge. However, well-structured multiple-choice testing can be successfully used to test more complex skills (Jones, 1997; Kolstad, 1994; Wilson, 1991). Research has suggested (Rhodes et al., 2004) that Multiple Choice Testing comprises a feasible assessment method, if the questions are qualitative in order to provoke students’ knowledge and understanding of teaching material.

According to Denenberg (1981), evaluation results, questioning and structure must all be based on quality; otherwise the assessment results are of little value. Multiple choice testing comprises a reliable evaluation method, not only in the theoretical field of Information Science but also in Programming. In addition, the test’s complexity could be increased in parallel with the increase of the number of suggested answers or with the addition of short-length answer questions.

Multiple choice questions are divided into two categories (Denenberg, 1981):

1. **Knowledge Questions**: they consist of questions on theoretical knowledge like gap-filling, true/ false and multiple choice.
2. **Programming ability questions**: they consist of code behavior questions to examine the capability of students to comprehend the logic of programming. More specifically, Denenberg (1981) suggests that students should be able to:
    - read a program (e.g. find the output of the program),
    - read a logical diagram (comprehension of its flows and operations),
    - convert a logical diagram to a code,
    - write a program (e.g. find commands from missing code).

Before exams are carried out, students should be fully informed on what they are supposed to do and how they are supposed to be graded.

Furthermore Traynor & Gibson (2005), determined the requirements for effective Multiple Choice Questions:

- **“Good Quality Code”: the code presented to the students should be of high standards. Unstructured code should not be used.**
- **“No tricks”: the questions should focus on the normal behavior of the programs.**
- **“Quality Distractors”: the erroneous answers given as alternatives should be appropriate and of high feasibility, so as to ensure the sense of correctness in answers.**

Next, some Educational software which are based on Assessment using Multiple Choice Questions are presented.

Traynor et al., (2006) developed an automated assessment system for Java Programming. They found a correlation between traditional and multiple choice testing methods. In both methods only well-prepared students succeeded. Parameterization of inputs determine the length, difficulty and the topic of the question. Similar code-behavior questions are presented by Lister and Leaney (2003b). The students must execute the program (trace) and select its behavior through 5 possible answers.

Brusilovsky and Sosnovsky, (2005) developed QuizPACK. Its aim was to produce and evaluate parameterized questions for student practice and assessment in Programming I. The student has to fill in the answer and hit the “submit” button. Then an immediate feedback of correct/erroneous answer is presented. There is Possibility of repeating a previous question. Each student is tested in different tests in Random question occurrence, through parameterization of the values of given[3]code variables. At the end a Final Score is reported.

Traynor and Gibson (2005) developed an intelligent MCQs Test. The students are called to find the output of a program. The questions are presented randomly and the students must choose one correct among five possible answers.

Lister (2005) developed a Multiple Choice Test to assess first semester students in Java Programming. The students are examined on thirteen questions of understanding the basics of Object-Oriented Programming concepts of classes, events and methods. Five questions involve comprehending basic structures such as sequence, condition, repetition and unknown code. Eight questions involve arrays including searching and sorting algorithms in known code prior to exam. The possible answers are 4 or 5 and the pass mark is both 50% and 70%. In the beginning the pass mark was 70% but as over half the class scored less than 18 out of 26, the pass mark was reduced to the traditional 50%.

Rhodes et al. (2004) developed ExamGen. A Multiple Choice question may contain 3, 4 or 5 possible answers. Questions are stored in a Microsoft Access Data Base. Also there is possibility of backward movement. Navigation is achieved through the Previous and Next buttons.

So, many researchers believe that Multiple-Choice Questions could be used of not only for the students’ assessment but also for students’ practice on basic knowledge of Programming. Moreover, the fact that correction and evaluation are carried out through the use of a computer renders the results objective and precise. For example, when a teacher has 100 papers to correct, there is the slight chance that s/he may over- or under-estimate somebody’s work. According to Habeshaw et al. (1992) the only way to examine students **with objectivity is by the use of Multiple Choice Questions**.

[3] The same code is used in multiple questions and simply the variable values are renewed.

3. Presentation of P.A.T. : Programming Assessment Testing

P.A.T., a Web-based fully automated assessment system was developed for the **Course of Application Development in a Programming Environment (Bakali et. al, 2004).** It was developed with the use of a Flash MX tool. The programming was conducted in ActionScript and the final files were extracted in html format. P.A.T. is not only a software to assess novice students in Programming but also it can predict their classification in National Exams. Programming comprises a core course in the Technological direction of the General Lykeion, as it is nationally examined for the admission to the Greek Universities (not necessarily only for Computer Science). This course is taught twice a week on a theoretical level and if there is enough time, students are encouraged to carry out practice training, i.e. code writing in a real programming environment[4] or other pedagogical software. Instructors assess students in two semesters[5]. The second semester tests include all the teaching material. Semesters tests and Panhellenic (National) exams[6] consist of paper-tests, involving True/ False, correspondence, output finding from a given code, conversion of logical diagrams into code or the opposite and code writing questions.

The emphasis concerning Semester tests or the Panhellenic (National) exams, is placed more on programming ability and knowledge questions (60%) than code writing (40%). It should be pointed out that students are examined in code writing only on paper. So, most of the students do not have the experience of solving problems in a real programming environment.

Since these students are novice programmers, the most effective assessment method involves the use of **Multiple Choice Questions** instead of Code Writing. As we have already mentioned, Code Writing requires for students to exhibit an advanced level of knowledge, in order to cope with the demanding material. Moreover, P.A.T. could be used in a **Summative Assessment** (Khamis, Indris, Ahmad and Indris, 2008) which could be used to assess the level of learning at the end of the course.

P.A.T. was used in the schools computer lab, under the supervision of the teaching staff, it takes approximately 45 minutes (one teaching hour). Students were assessed on 30 questions at the end of 2nd Semester and before the Panhellenic (National) exams. The students could use P.A.T. in the schools' computer laboratory or via Web from their home or elsewhere.

During May 2009, 73 students from two schools (43 students from 1st unit and 30 students from 2nd unit) used P.A.T.. Also they answered evaluation Questionnaires regarding P.A.T.'s Environment, Question Content and Usefulness. Results show that 61 students out of 73 found the experience positive and the tool very useful to increase their depth of knowledge in programming course and that they have been helped to discover their shortcomings. However most of them think that they were under estimated by P.A.T. comparison to traditional exams. In our opinion this

[4] They use a pseudo-language named "Glossa" which can be best described as a Greek translation of Pascal.

[5] At the end of the year the average between first and second semester is computed which determines the final grade for this lesson in the school certificate.

[6] These exams determine if the students are going to continue their studies in a High Educational Institution (University or Technological Educational Institution).

happens because they do not have the experience in computerized exams. Most of the students (especially, low performance students) preferred to use P.A.T. for learning and self-assessment than for testing.

Below are presented the reasons for using P.A.T.

- Students will be able to practice and be assessed in Knowledge and Programming Ability Questions.
- Most of the teachers who teach the programming course complain about the fact that teaching hours suffice only for teaching the exams material, leaving little time for practice. Through P.A.T. students will be able to practice more frequently, not just in the laboratory environment but also via the Web.
- Through the use of P.A.T., students will be able to discover their shortcomings in order to be prepared for the National Exams.
- P.A.T.'s friendliness will attract students of all levels to participate and practice as frequently as possible in order to increase their programming skills.

4. Questions in P.A.T.

The book's structure is such, so that the exam material is repeated (Bakali et. al, 2004). Chapters 1, 4 and 6 are theoretical and serve as an introduction on the necessity of Programming; Chapter 7 refers to the basic Programming elements and a presents the pseudo-language (GLOSSA); chapters 2 and 8 are an introduction to the structure of Sequence, Choice and Repetition; chapters 3 and 9 present Data Structures, with an emphasis on Tables; finally, chapter 10 deals with Sub-Programs.

Each question belong to a difficulty level: A = easy question, B = moderate question, C = difficult question. The content of questions in P.A.T. was designed according to the low levels of Bloom's Taxonomy (Bloom, 1956).

The following Categories of Questions were developed:

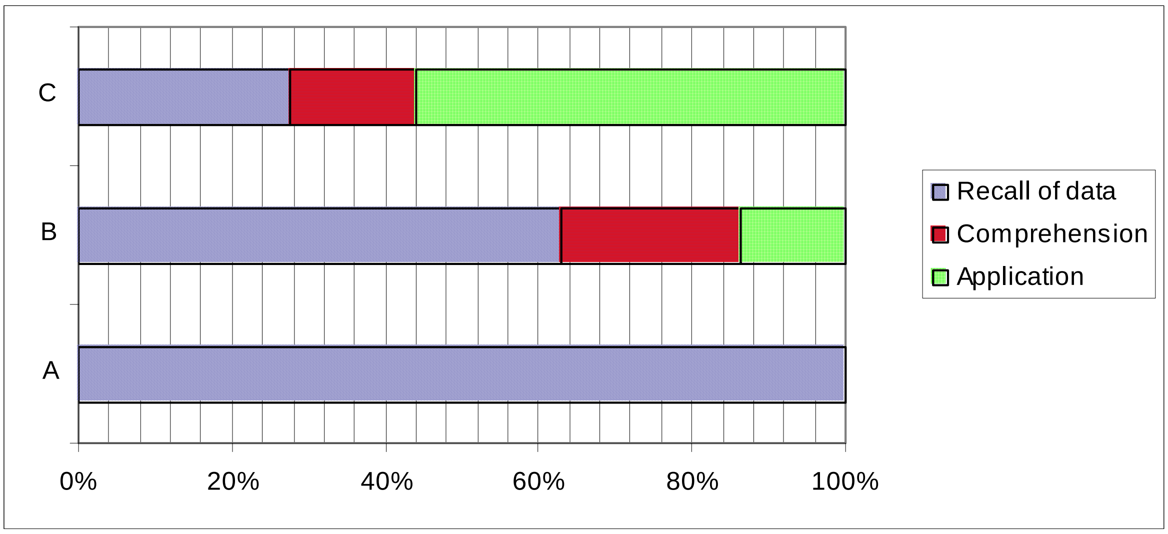
- **Recall of data** : Knowledge questions on the Theory of the course, the Syntax and Function of Frameworks of Structured Programming and of Sub-Programs in True/False and Multiple Choice Question format (Level A, B and C). Such questions examine students' memorization capability.
- **Comprehension**: A piece of code and a question involving the behavior of the code (finding the output after the execution of a program). Such questions have been found efficient (Lister, 2001) as far as students assessment on their ability to read and comprehend the code's Semantic (level B and C).
- **Application**: Exercises to examine students' skills to apply prior knowledge to new problems. Three types of exercises were used; 1) A piece of code, which can be realized through a Structure of Process or Choice or Repetition, where a student is called to choose an equivalent command for the execution of the above functions (level B). 2) Also a Logical Diagram is given, where the student is called upon to find the equivalent command to express one or more functions (level C). 3) Gap filling in a piece of code or program according to some expressions (Lister and Leaney, 2003a). Program gap filling (level B and mostly level C) is the most difficult activity and needs much more consideration and capabilities, also it helps students in increasing their power of solving sub-problems (Hwang et al., 2008).

The students were not examined at high levels of Blooms' Taxonomy. Oliver and Dobele (2007) showed that the pass rates of courses with higher cognitive demands (Analysis, Synthesis and Evaluation) were increased in relation with lower cognitive demands (Recall of data, Comprehension and Application). This means that if first year experience in programming has a high cognitive level of assessment then weaker students are prevented to continue their studies in this science.

The following **Table 1** and **Graph 1** show the number of questions which respect to Blooms' Taxonomy and difficulty level.

	A	B	C	Total Questions per Blooms' Taxonomy
Recall of data	186	93	30	**309**
Comprehension	0	35	18	**53**
Application	0	20	61	**81**
Total Questions per difficulty level	**186**	**148**	**109**	**443**

Table 1: Number of questions which respect to Blooms' Taxonomy and difficulty level.



Graph 1: Percentages of questions which respect to Blooms' Taxonomy and difficulty level.

Another criterion which increases the difficulty of question and demands of the depth of knowledge is the number of possible answers. Table2 presents the number of possible answers per difficulty level.

Level of Questions	**Possible Answers in Multiple Choice Questions**	**True/ False Questions (2 possible answers)**
A	3	EXIST
B	4	EXIST
C	5	NOT EXIST

Table2: Number of possible answers per difficulty level

4.1 Model Structure

- A random presentation of 30 questions from all Chapters of the exam material, depending on the students' level. Each student is tested on different questions at different levels. This ensures the quality of the exams as far as cheating is concerned, since students sit in close proximity in computer laboratories.
- The student moves from one difficulty level to another according to his/her answer (Figure 1). At the end of the test, the total number of correct answers per level, in relation to the total number of questions presented per level, is shown.
- Questions at level A test students on knowledge. A correct answer to a question A leads to question at level B, which includes Programming Ability Questions too. If student answers correctly level B question then question from level C is appeared. Students at level C are tested mostly in Programming Ability Questions.
- Question counter per chapter.
- Wrong answer counter per chapter.
- At the end of the test, the Total number of correct answers out of 30 and the students' classification are presented.
- Also a Final Score, which depends on the level of questions correctly answered, is presented.

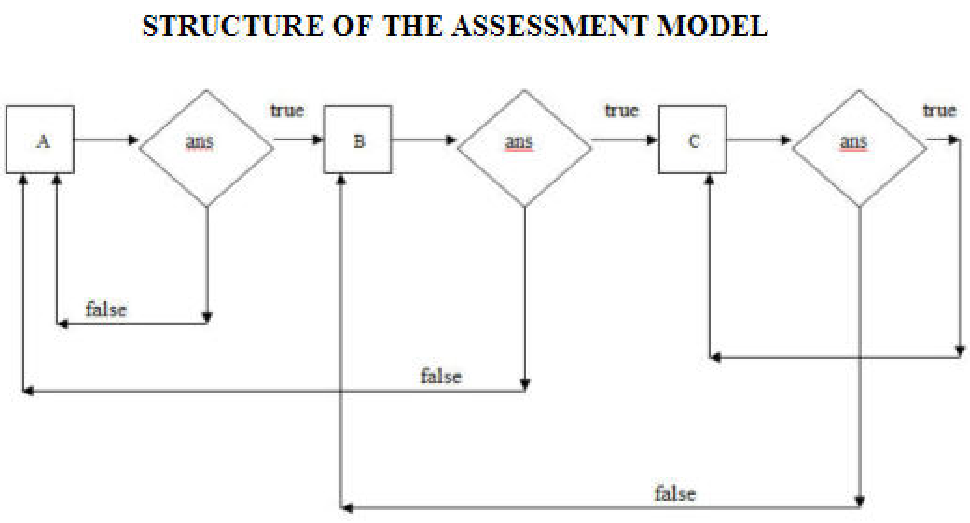


Figure 1:
Adaptive Sequence of question in P.A.T.

5. Results

Significant weight was placed on Feedback. P.A.T. seeks to serve both the teacher and the student. As far as the student is concerned, P.A.T. not only serves as a means of practice on the exam material, but also as a means of feedback on his/her shortcomings per chapter. As far as the teacher is concerned, P.A.T. functions as a means of assessing the students' Level which indicates how well they are prepared for Panhellenic (National) exams and if it is possible to help them to overcome their weakness until then.

5.1 Analysis of the results from the teachers' point of view

If, following the aforementioned structure (Figure 1), the student correctly answers all 30 questions (from 0 to 29), s/he will obtain the following best performance sequence

A, B, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C.

On the contrary, if the student answers all 30 questions incorrectly, the worst performance sequence will be as follows:

A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A,A.

In the Results printout, the answers given by the student will be characterized by the letter of the level and the corresponding question number, **LQn,** where **L** is the difficulty level (**A, B or C**) and **Qn** is the number of the question at the corresponding Level (**Qn= 0..185 for Level A**, **Qn=0..147 for Level B and Qn=0..108 for Level C**). For example:

A5, B7, C3, B33, A12, B1, C77, C4, C100, B18, C5, C7, B22, A23, A27, A34, A47, A61, B75, C62, C55, C59, B81, C80, C19, B9, C0, C41, B29, A30.

This Questions' sequence helps the teacher to immediately recognize which questions the student failed. Regarding the example's questions sequence, the student answered wrongly the following questions:

C3: because a Level B question follows
B33: because a Level A question follows
Also C100, C7, B22, A23, A27, A34, A47, C59, C19, C41 and B29.

At the end of the test, the following results are presented for each student:
(a) **Total Results**: Number of the correct answers out of 30, (b) **Number of the Correct Answers per Level in relation to the total number of questions per Level,** (c) **Final Score** given by the following formula :

Final Score = 1* Number of Correct Answers at Level A+
2* Number of Correct Answers at Level B+
3* Number of Correct Answers at Level C

(d) Classification of student which depends both of the **Final Score** and the **Total Result**

Based on our research using 73 students we present 3 classifications:

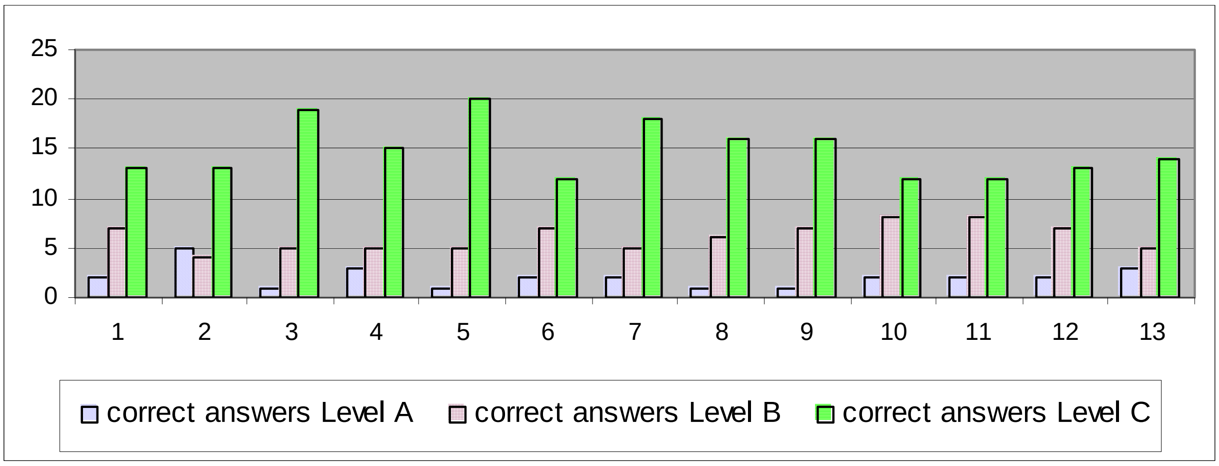
5.1.1 High Programming Skills' students

We consider that student could be classified as High Programming Skills' student if s/he answered, at least correctly 21 questions and obtain Final Score at least 52/87 (60%). If the student answers all questions correctly, the question sequence will be:
A B C C C C C C C C C C C C C C C C C C C C C C C C C C C C

s/he achieves the following:
Final Score = 1*1 + 1*2 + 28*3 = 87/87

and the corresponding **Total Result** will be **30/30**.

A High Programming Skills' student will answer correctly questions mostly at Level C **(Graph 2)**.



Graph 2: Correct answers per Level by High Programming Skills' students (13 students out of 73)

In order to support our argument for High Programming Skills' students, the following Table 3 is presented:

	Mean	StDev
correct answers from Level A	2,077	1,115
correct answers from Level B	6,077	1,320
correct answers from Level C	14,846	2,764

Table 3: Correct answers per Level by the 13 High Programming Skills' students

where Mean is the average number of correct answers per High Programming Skills student and StDev is the standard deviation of the number of correct answers per student. As we can observe from Table 3 these students answered correctly in average 15 out of 30 (50%) questions from Level C.

<u>Example 1-High Programming Skills' student with the lowest Total Result and Final Score</u>

A **A B** C B A **A B** C **B C C C C** C **B C** C **B C C C** C **B C** C **B C C C**

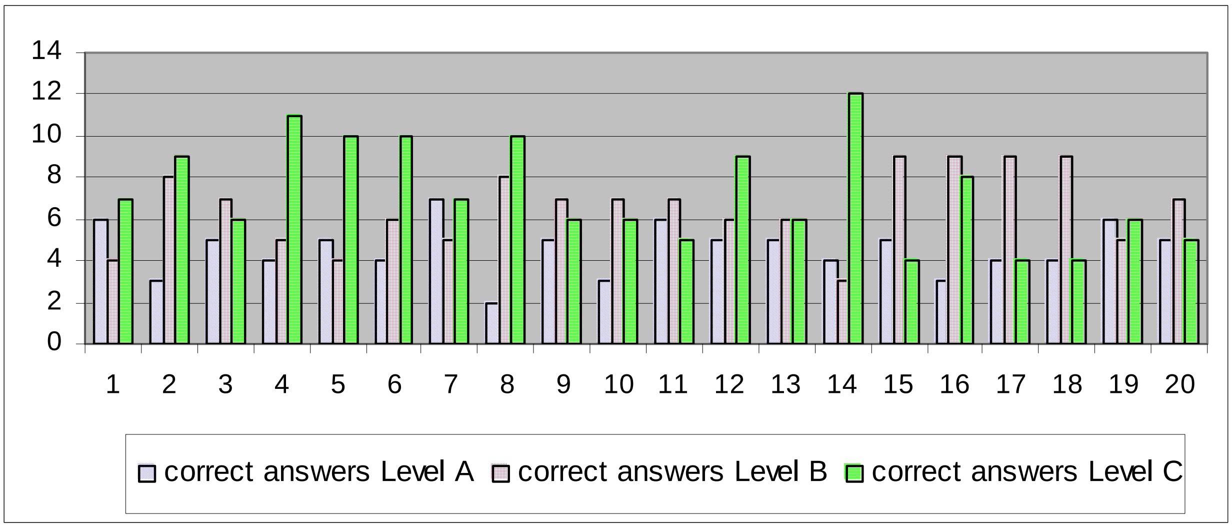
The student has 2 correct answers on Level A, 7 correct answers on Level B and 12 on Level C. His/her
Final Score = 2*1 + 7*2 + 12*3 = 2 + 14 + 36 = 52/87,

and the corresponding **Total Result** = **21/30**.

The majority of answers correctly answered (12) belong to Level C.

5.1.2 Medium Programming Skills' students

If a student performed well in Knowledge Questions and at a moderate level in Programming ability Questions, s/he will be classified as a Medium Programming Skills student. In our sample most of the students answered correctly questions mostly to Level B and C (**Graph 3**).
In order for the student to be classified as a Medium Programming Skills' student s/he will have to score at least **34/87 (39%)** and achieve a **Total Result** of at least **16/30**. The highest grade for a Medium Programming Skills' student is **20/30** for the **Total Result** and **51/87 (58,6%)** for the **Final Score**.



Graph 3: Correct answers per Level by Medium Programming Skills' students (20 students out of 73)

In order to support our argument for Medium Programming Skills' students, the following Table 4 is presented:

	Mean	StDev
correct answers from Level A	4,55	1,234
correct answers from Level B	6,55	1,82
correct answers from Level C	7,25	2,468

Table 4: Correct answers per Level by the 20 Medium Programming Skills' students

As we can see from the table the number of correct answers is spread across all Levels but the majority of them are from Level B and C (14 out of 30, approximately 50%).

<u>Example 1- Medium Programming Skills' student with most correct answers from level B</u>

A **A B C** C **B C** C **B A** B **A B C** C B A **A B** C **B C C C** C B **A B** C B

The student has 5 correct answers on Level A, 7 correct answers on Level B and 6 on Level C. His/her

Final Score = 5*1 + 7*2 + 6*3 = 5 + 14 + 18 = 37/87,

and the corresponding **Total Result** = **18/30**.

Example 2- Medium Programming Skills’ student with most correct answers from level C

A **A B** C **B** C **B C** B A **B C C** C B **A B** C B **A B C C C C C C C C** C

The student has 4 correct answers on Level A, 6 on Level B and 10 on Level C. His/her

Final Score = 4*1 + 6*2 + 10*3 = 4 + 12 + 30 = 46/87,

and the corresponding **Total Result** = **20/30**.

This means that the majority of answers correctly answered (10) belong to Level C but this is not enough to place the student at High Programming Skills. As we can observe the student answers wrongly 10 out of 30 questions and as a result s/he is properly placed as a Medium Programming Skills’ student.

Example 3- Medium Programming Skills’ student with the lowest Final Score

A B C B A **A B** C **B C C** C B **A B** C B **A B** C **B** C **B C C** C **B** C **B** C

The student has 4 correct answers on Level A, 9 correct answers on Level B and 4 on Level C. His/her

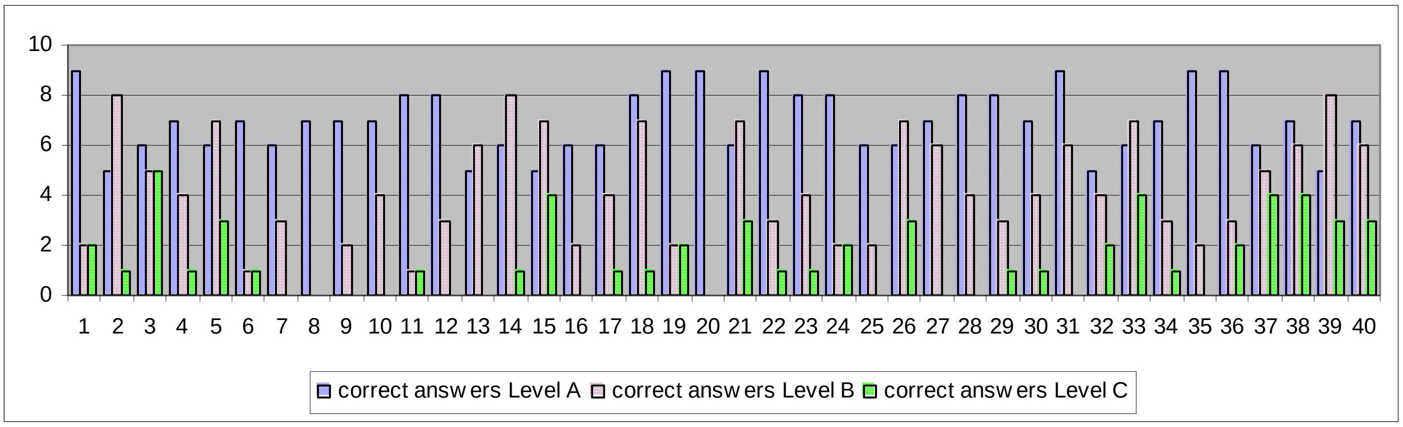
Final Score = 4*1 + 9*2 + 4*3 = 4 + 18 + 12 = 34/87,

and the corresponding **Total Result** = **17/30**.

5.1.3 Low Programming Skills’ students

A Low Programming Skills’ student needs to study more. The majority of his /her correct answers do not necessarily belong to level A. However, the percentage of level C correct answers must be lower than that of Level A and B. Otherwise the student has problem in questions that requires memorization.

Nevertheless most of the students’ correct answers were from Level A (recall of data), 31 students out of 40 show that frequency (**Graph 4**).



Graph 4: Correct answers per Level by Low Programming Skills' students (40 students out of 73)

The highest **Total Result** that can be achieved is **17/30.** Also the highest **Final Score** is **33/87**.

In order to support our argument for Low Programming Skills' student, the Table 5 is presented:

	Mean	StDev
correct answers from Level A	7	1,301
correct answers from Level B	4,2	2,301
correct answers from Level C	1,45	1,449

Table 5: Correct answers per Level by the 40 Low Programming Skills' students

The above table (Table 4) shows that the majority of correct answers are mostly from Level A. Also the average number of correct answers from Level C is poor performed.

<u>Example 1 -Low Programming Skills' student with most correct answers from level A</u>

A A A A A A **A** B **A** B A **A** B A A A A A **A** B A **A** B A A **A** B A **A** B

This student only has 7 correct answers at Level A. His/her

Final Score = 1*7 = 7/87

and the corresponding **Total Result** = **7/30.**

Example 2-Low Programming Skills' student with most correct answers from level B

A A **A B** C B A A **A** B **A B** C **B** C **B C** C **B** C **B** C **B** C B **A B** A **A** B

The student has 5 correct answers on Level A, 8 correct answers on Level B and 1 on Level C. His/her

Final Score = 5*1 + 8*2 + 1*3 = 5 + 16 + 3 = 24/87

and the corresponding **Total Result** = **14/30**. This student has answered correctly most of the questions from level A (5/10) and B (8/12) but s/he answered correctly only 1 question out of 8 from level C.

Finally, the Application Menu also includes the choice "teacher". The teacher can provide an appropriate user name and password and have the ability to read or print all the questions according to level and per chapter. Through this choice, the teacher can evaluate the students' shortcomings in detail (which questions and what chapters).

5.2 Analysis of the results from the students' point of view

Upon completion of the 30-question test, the result section produces the following: (a) Total Result (b) Final Score (c) Classification and (d) Analytical Results. More specifically:

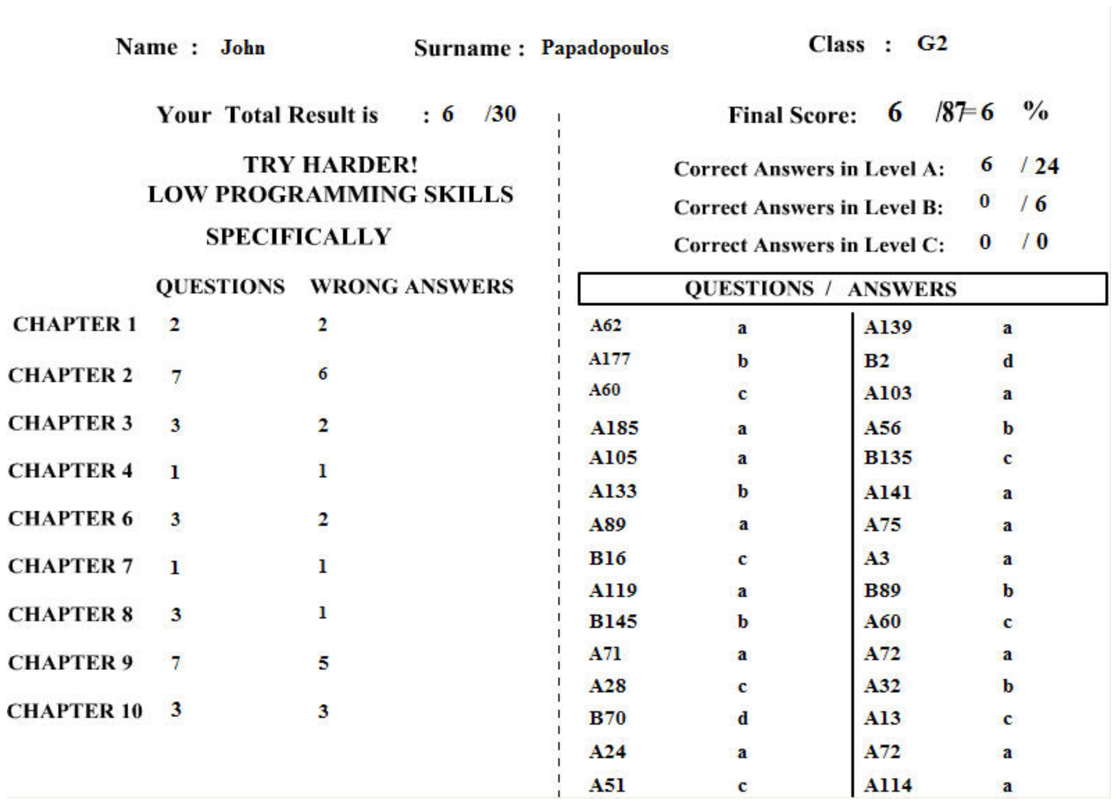
(a) **Total Result**: number of correct answers out of 30 (x/30).
(b) **Final Score**: it depends on the number of Correct Answers per questions level.
(c) **Classification:** according to the Total Result (x) and the Final Score (y).

```
if  (0<=x<=17) and (0<=y<=33) →
    TRY HARDER - LOW PROGRAMMING SKILLS!
if (16<=x<=20) and (34<=y<=51)→
     GOOD – MODERATE PROGRAMMING SKILLS!!
if (21<=x<=30) and (52<=y<=87)→
     VERY GOOD – HIGH PROGRAMMING SKILLS!!!
```

In order for a student **to have excellent results**, his/her **Total Result** must be at least **21 out of 30**, with the characterization **"VERY GOOD"/ HIGH PROGRAMMING SKILLS** and his/her **Final Score** must be at least **52/87 (60%)**. Furthermore a satisfied result in order for a student **to be successful**, his/her **Total Result** must between **16** to **20 out of 30**, with the characterization , **"GOOD"/ MODERATE PROGRAMMING SKILLS** and his/her **Final Score** must be between **34 (39%)** to **51/87**.

(d) **Analytical Results**: this section contains all Questions presented to the student per chapter during the test and the Total Incorrect Answers per chapter. This facilitates the student's feedback process, as s/he can study again the chapters with the most incorrect answers.

An example of a Result Section template is presented (Figure 2):



Name : John Surname : Papadopoulos Class : G2

Your Total Result is : 6 /30

Final Score: 6 /87=6 %

TRY HARDER!
LOW PROGRAMMING SKILLS

SPECIFICALLY

	QUESTIONS	WRONG ANSWERS
CHAPTER 1	2	2
CHAPTER 2	7	6
CHAPTER 3	3	2
CHAPTER 4	1	1
CHAPTER 6	3	2
CHAPTER 7	1	1
CHAPTER 8	3	1
CHAPTER 9	7	5
CHAPTER 10	3	3

Correct Answers in Level A:	6	/ 24
Correct Answers in Level B:	0	/ 6
Correct Answers in Level C:	0	/ 0

QUESTIONS /	ANSWERS		
A62	a	A139	a
A177	b	B2	d
A60	c	A103	a
A185	a	A56	b
A105	a	B135	c
A133	b	A141	a
A89	a	A75	a
B16	c	A3	a
A119	a	B89	b
B145	b	A60	c
A71	a	A72	a
A28	c	A32	b
B70	d	A13	c
A24	a	A72	a
A51	c	A114	a

Figure 2
RESULT TEMPLATE : TRY HARDER/ LOW PROGRAMMING ABILITY

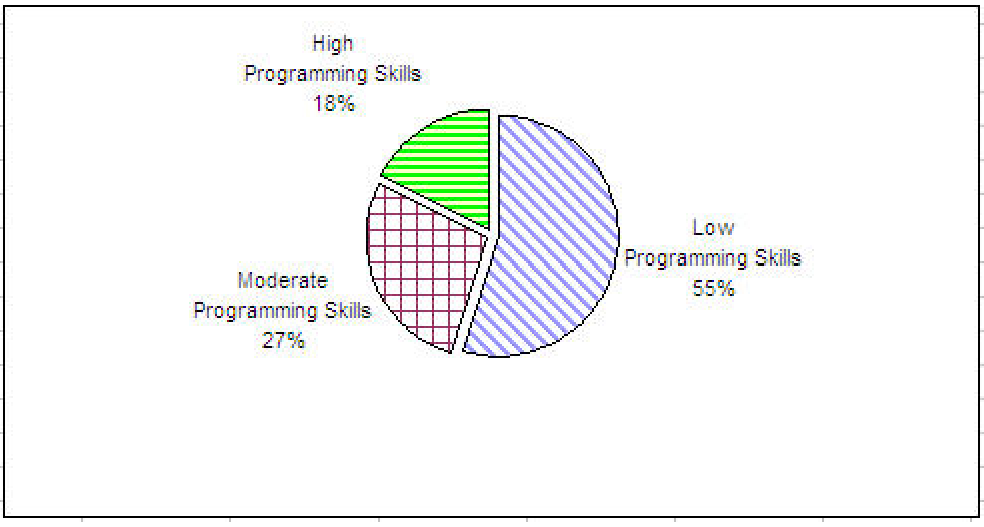
Upon closer observation of the Result Section template, it can be inferred that the majority of the student's correct answers belong to Level A (6 Questions: A89, A119, A28, A139, A56, A3) and his **Total Result** is 6 correct questions out of 30 questions examined in total. So, the student was unsuccessful in most of questions.

The student's **Final Score** is 6/87 (6%). More specifically, of the 24 Level A Questions he answered correctly only 6.

Final Score = 6*1 + 0*2 + 0*3 = 6 + 0 + 0 = 6 out of 87
It is obvious that he is a Low Programming Skills student.

6. Prediction of students' classification in National Exams

The results of 73 students that took the test on P.A.T. (**Graph 5**) indicate that 45% performed well. However, 55% of them need to practice more in order to achieve better grade in Panhellenic (National) exams (Low Programming Skills). Most of the students do not practice often. They memorize instead of comprehending the logic of programming.



Graph 5: Classification of students using P.A.T.

The relation between P.A.T. classification and their performance in National Exams is presented in the following Table 5.

Programming Skills	Total Result	Final Score	Performance in National Exams
High	21-30	52-87	18-20
Medium	16-20	34-51	12-17,9
Low	0-17	0-33	0- 11,9

Table 6: Correspondence between P.A.T. classification, Total Result, Final Score and Performance in National Exams

Using P.A.T. classification, in the two high schools where this study was carried out, we predicted that in the 2009 – National Exams, 55% of students will score below 11,9, 27% between 12 and 17,9 and 18% between 18 and 20.

The **Reliability and Objectivity of P.A.T**. is confirmed when we compare the expected performance with the results in the National Exams (Table 7).

Programming Skills	Performance in National Exams	P.A.T. classification		National Exams classification	
High	18-20	18%	**45%**	17%	**44%**
Medium	12-17,9	27%		27%	
Low	0- 11,9		**55%**		**56%**

Table 7: Correspondence between P.A.T. Assessment and National Exams Assessment

7. Strengths and Weakness of P.A.T.

Strengths

- Adaptation to the students' programming skills.
- Successful classification of students.
- Prediction of students' performance in National Exams.
- Efficiency, Objectivity and Reliability.
- Automated Assessment Process.
- Speed in Results production.
- Large library of questions, possibility of Test repetition with renewed interest. It contains **443 questions.**
- Memorization of questions by students is rendered difficult.
- Indication of students' sufficient preparation for participation in Panhellenic (National) exams.
- Exposure of students' weaknesses per chapter of the exam curriculum.
- Pleasant and usable Graphic Work Environment (it was developed with FlashMx).
- Convenience of practice in school laboratories (local), but also the Web.
- The execution of P.A.T. software requires only the installation of a browser and one can run the program from any hard disk device even without Internet Connection.

Weaknesses

- P.A.T. was developed to test novice programmers, only.
- P.A.T. was developed to test student's programming skills on "Glossa", a pseudo-language for Greek students.

8. Conclusions and Future Goals

P.A.T. is not only an Assessment tool but also it can predict the students' classification in National Exams too. Different schools in different countries have different requirements for teaching computer programming. We developed P.A.T., for helping Greek high school students and teachers to evaluate students' programming skills. P.A.T. could be executed so that a student can choose the chapter and the level that s/he wishes to be examined. So P.A.T. could be used as a Learning tool too. Also, it could enable the teachers to upload their own questions. In addition, it would be interesting if the system produces some statistics results about students' performance which will be available to both students and teachers. At the end another achievement would be students be able to practice in simple code writing exercises too.

APPENDIX

A) EXAMPLES OF P.A.T. QUESTIONS

1) LEVEL C – DIFFICULT QUESTIONS

EXAMPLE 1 – KNOWLEDGE

C53) Με τον όρο Οδηγούμενη από το Γεγονός Προγραμματισμό εννοούμε

a) Τη δυνατότητα δημιουργίας Βάσεων Δεδομένων.

b) Τη δυνατότητα να δημιουργούμε γραφικά ολόκληρο το περιβάλλον της Εφαρμογής.

c) Τη δυνατότητα δημιουργίας Δικτυακών Εφαρμογών.

d) Τη δυνατότητα να ενεργοποιούνται λειτουργίες του Προγράμματος με την εκτέλεση ενός Γεγονότος.

e) Τη δυνατότητα δημιουργίας Εμπορικών Εφαρμογών.

EXAMPLE 2 – COMPREHENSION

C18) Τι θα τυπώσει ο παρακάτω Αλγόριθμος για x= -1;

```
Αλγόριθμος Συνάρτηση
Διάβασε x
Αν x<1 τότε
fx<-(16/(x-1)^2)div 2
Αλλιώς_αν x=1 τότε
fx<-2
Αλλιώς
fx<-16/(x-1)^3
Τέλος_αν
Τέλος Συνάρτηση
```

a) -1

b) -4

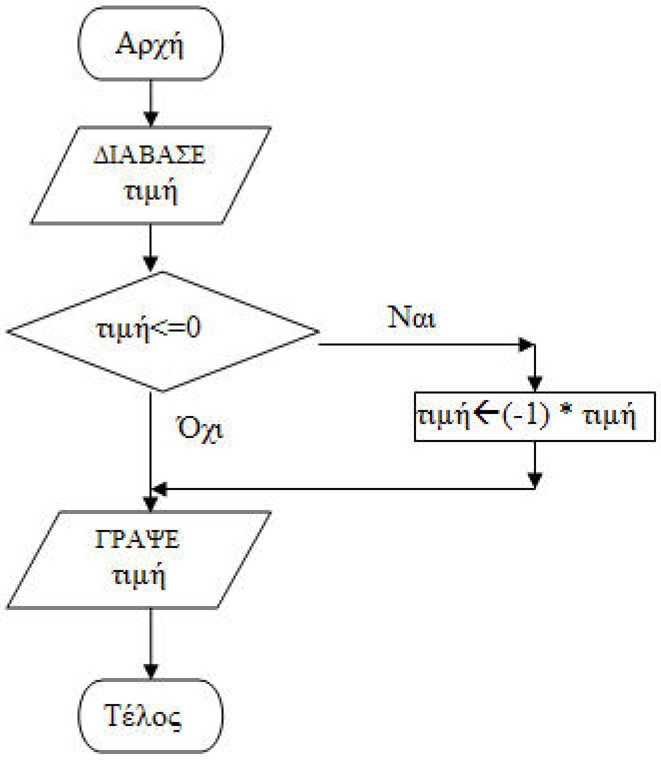
c) -2

d) 1

e) 2

EXAMPLE 3 – APPLICATION

C70)Με ποια από τις παρακάτω εντολές υλοποιείται η Δομή της Επιλογής του Λογικού Διαγράμματος που σας δίνεται;



a)ΑΝ τιμή <= 0 ΤΟΤΕ τιμή <-(-1) * τιμή ΓΡΑΨΕ τιμή ΤΕΛΟΣ_ΑΝ

b)ΑΝ τιμή <= 0 ΤΟΤΕ τιμή <-(-1) * τιμή ΤΕΛΟΣ_ΑΝ

c)ΑΝ τιμή <= 0 ΤΟΤΕ τιμή <-(-1) * τιμή ΑΛΛΙΩΣ ΓΡΑΨΕ τιμή ΤΕΛΟΣ_ΑΝ

d)ΕΠΙΛΕΞΕ τιμή ΠΕΡΙΠΤΩΣΗ <=0 τιμή <-(-1) * τιμή ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ

e)ΑΝ τιμή > 0 ΤΟΤΕ τιμή <-(-1) * τιμή ΤΕΛΟΣ_ΑΝ

EXAMPLE 4 – APPLICATION

C94)Συμπλήρωσε τα κενά στο παρακάτω Πρόγραμμα, το οποίο διαβάζει 2 ακέραιους και υπολογίζει το Μέγιστο Κοινό Διαιρέτη με την χρήση της Συνάρτησης.

```
ΠΡΟΓΡΑΜΜΑ ΜΚΔ1
ΜΕΤΑΒΛΗΤΕΣ
ΑΚΕΡΑΙΕΣ:Α, Β, ΜΚΔ
ΑΡΧΗ
ΓΡΑΨΕ ' ΔΩΣΕ ΤΟΥΣ 2 ΑΡΙΘΜΟΥΣ'
ΔΙΑΒΑΣΕ Α, Β
ΜΚΔ<-  (1)
ΓΡΑΨΕ 'ΜΕΓΙΣΤΟΣ ΚΟΙΝΟΣ ΔΙΑΙΡΕΤΗΣ:', ΜΚΔ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ ΜΚΔ1


ΣΥΝΑΡΤΗΣΗ (2) (ΑΡ1, ΑΡ2): (3)
ΜΕΤΑΒΛΗΤΕΣ
ΑΚΕΡΑΙΕΣ:ΑΡ1, ΑΡ2, ΠΗΛ, ΥΠΟΛ
ΑΡΧΗ
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
ΠΗΛ<-ΑΡ1 DIV ΑΡ2
ΥΠΟΛ<- ΑΡ1 MOD ΑΡ2
ΑΝ (4) ΤΟΤΕ
ΑΡ1<- ΑΡ2
ΑΡ2<-ΥΠΟΛ
ΤΕΛΟΣ_ΑΝ
ΜΕΧΡΙΣ_ΟΤΟΥ ΥΠΟΛ=0
ΥΠ_ΜΚΔ<- (5)
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

a)(1)=ΥΠ_ΜΚΔ(Α,Β),(2)=ΥΠ_ΜΚΔ ,(3)=ΑΚΕΡΑΙΑ,(4)=ΥΠΟΛ <>0,(5)=ΑΡ2

b)(1)=ΥΠ_ΜΚΔ(ΑΡ1,ΑΡ2),(2)=ΥΠ_ΜΚΔ ,(3)=ΠΡΑΓΜΑΤΙΚΗ,(4)=ΥΠΟΛ <>0,(5)=ΑΡ1

c)(1)=ΥΠ_ΜΚΔ(Α,Β),(2)=ΜΚΔ ,(3)=ΑΚΕΡΑΙΑ,(4)=ΥΠΟΛ <>1,(5)=ΑΡ2

d)(1)=ΜΚΔ(Α,Β),(2)=ΥΠ_ΜΚΔ ,(3)=ΑΚΕΡΑΙΑ,(4)=ΥΠΟΛ <>0,(5)=ΑΡ1

e)(1)=ΜΚΔ(Α,Β),(2)=ΜΚΔ ,(3)=ΠΡΑΓΜΑΤΙΚΗ,(4)=ΥΠΟΛ <>0,(5)=ΥΠΟΛ

2)LEVEL B – MODERATE QUESTIONS

EXAMPLE 1 – KNOWLEDGE

B93)Το αποτέλεσμα της εντολής εκχώρησης είναι

a) Η αντικατάσταση της τρέχουσας τιμής της μεταβλητής (δεξιά) με την τιμή της παράστασης που υπάρχει αριστερά.

b) Η αντικατάσταση της τρέχουσας τιμής της μεταβλητής (αριστερά) με την τιμή της παράστασης που υπάρχει δεξιά.

c) Η αύξηση της τρέχουσας τιμής της μεταβλητής (αριστερά) με την τιμή της παράστασης που υπάρχει δεξιά.

d) Η αύξηση της τρέχουσας τιμής της μεταβλητής (δεξιά) με την τιμή της παράστασης που υπάρχει αριστερά.

EXAMPLE 2 – COMPREHENSION

B58)Τι θα τυπώσει το παρακάτω τμήμα Αλγορίθμου για κ=3;

```
Διάβασε κ
Για i απο 1 μέχρι 3
A[i]<-i+2
Τέλος_επανάληψης
Για i απο 2 μέχρι 2
Αν A[i]=κ τότε Εκτύπωσε "A"
Αλλιώς_αν A[i]=κ+1 τότε Εκτύπωσε "B"
Αλλιώς_αν A[i]=κ+2 Εκτύπωσε "C"
Τέλος_αν
Τέλος_επανάληψης
```

a) A , B

b) A

c) C

d) B

EXAMPLE 3 – APPLICATION

B107)Πως αλλιώς μπορούν να διατυπωθούν η παρακάτω εντολές (σχετικό παράδειγμα στο βιβλίο σελ. 38);

```
ΑΝ Βάρος < 80 ΤΟΤΕ
ΑΝ Ύψος < 1.70 ΤΟΤΕ ΓΡΑΨΕ 'ελαφρύς - κοντός'
ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΑΝ
```

a) ΑΝ (Βάρος < 80) ΚΑΙ (Ύψος < 1.70) ΤΟΤΕ
ΓΡΑΨΕ 'ελαφρύς - κοντός' ΤΕΛΟΣ_ΑΝ

b) ΑΝ (Βάρος < 80) Ή (Ύψος < 1.70) ΤΟΤΕ
ΓΡΑΨΕ 'ελαφρύς - κοντός' ΤΕΛΟΣ_ΑΝ

c) ΑΝ Βάρος < 80 ΤΟΤΕ ΓΡΑΨΕ 'ελαφρύς' ΤΕΛΟΣ_ΑΝ
ΑΝ Ύψος < 1.70 ΤΟΤΕΓΡΑΨΕ 'κοντός'ΤΕΛΟΣ_ΑΝ

d) ΑΝ (Βάρος < 80) ΤΟΤΕ ΓΡΑΨΕ 'ελαφρύς - κοντός'
ΤΕΛΟΣ_ΑΝ

3) LEVEL A – EASY QUESTIONS

EXAMPLE 1 – KNOWLEDGE

A2)Ποια είναι τα στάδια αντιμετώπισης ενός προβλήματος;

a) Ανάλυση και Επίλυση.

b) Κατανόηση, Ανάλυση και Επίλυση.

c) Κατανόηση και Επίλυση.

EXAMPLE 2 – KNOWLEDGE

A24)Οι τελεστές +, -, *, /,^ ονομάζονται

a) Λογικοί.

b) Συγκριτικοί.

c) Αριθμητικοί.

EXAMPLE 3 – KNOWLEDGE

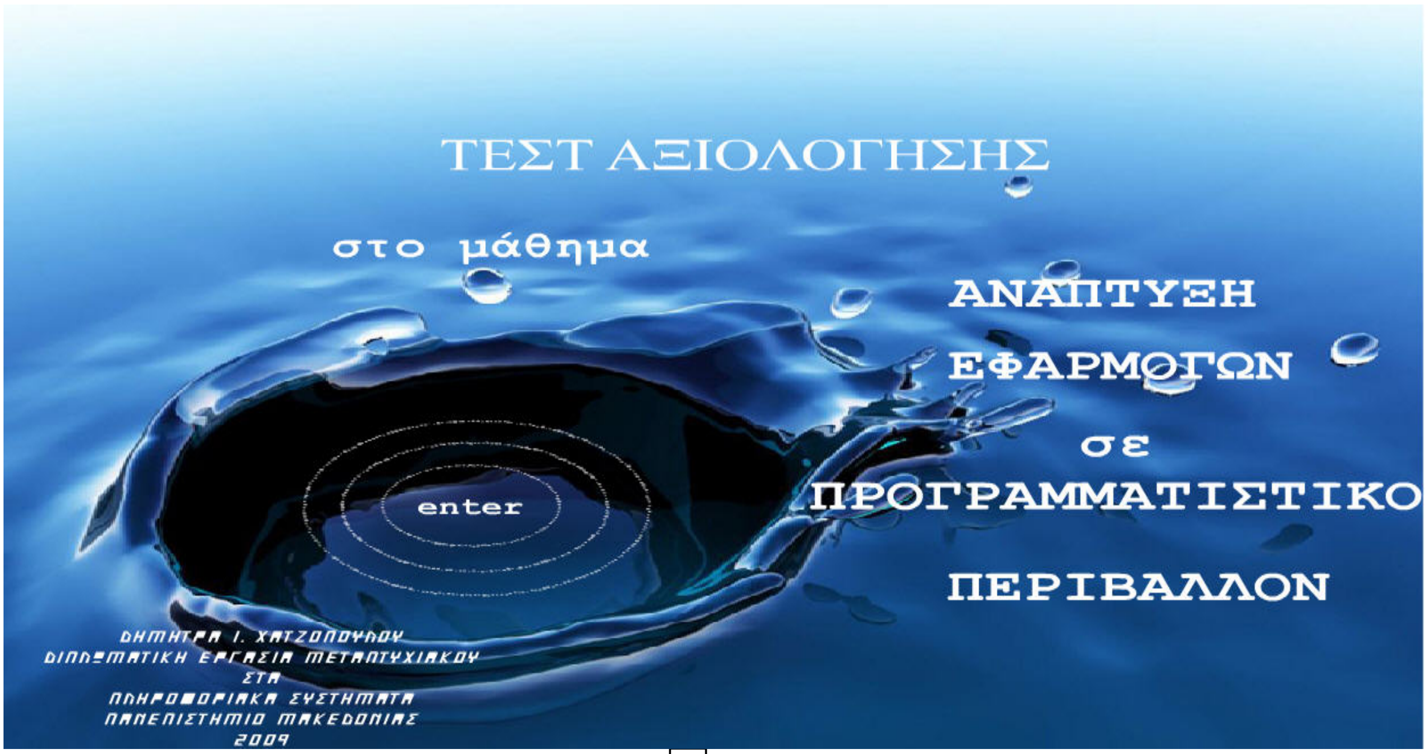
A52)Υπάρχει Αλγόριθμος για την Σχεδίαση Αλγορίθμων;

a) Αληθής

b) Ψευδής

B) PRESENTATION OF P.A.T.

B1) INTRO



ΤΕΣΤ ΑΞΙΟΛΟΓΗΣΗΣ

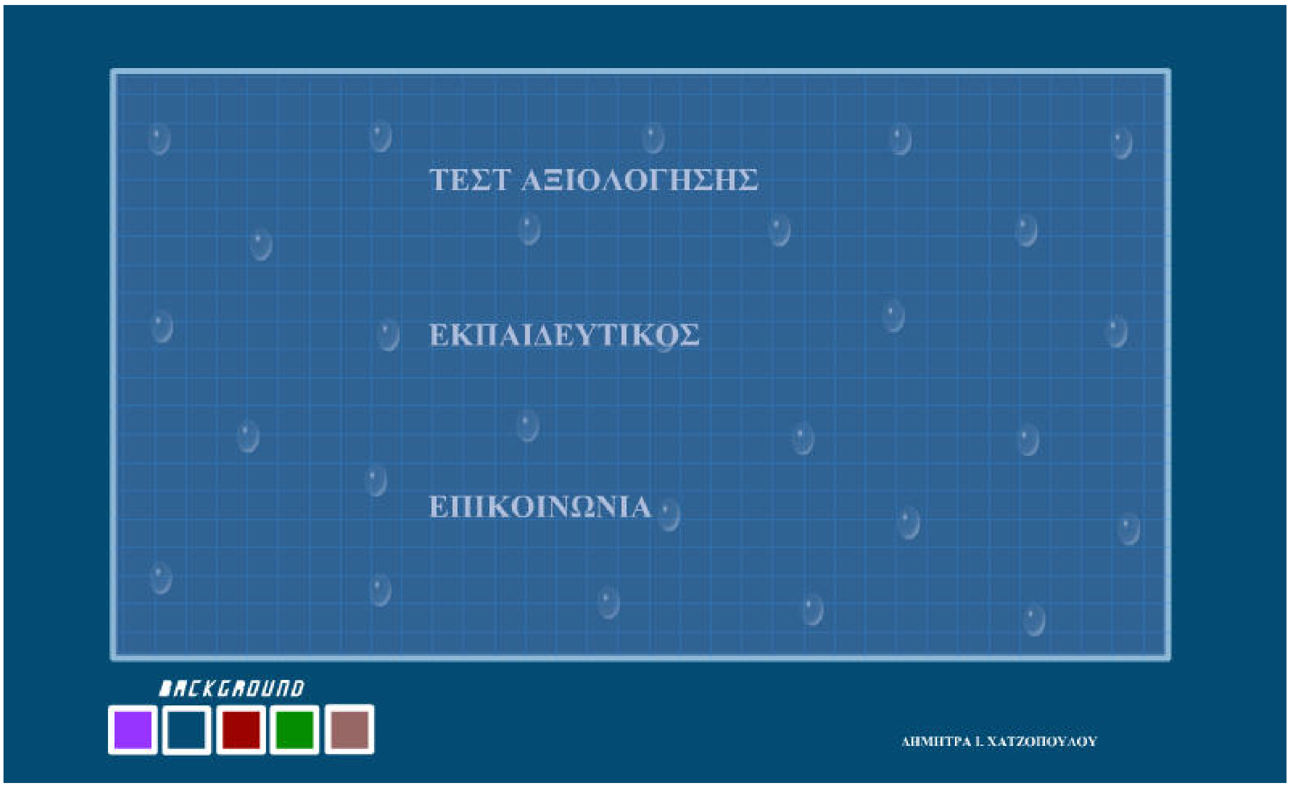
στο μάθημα

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ σε ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΟ ΠΕΡΙΒΑΛΛΟΝ

enter

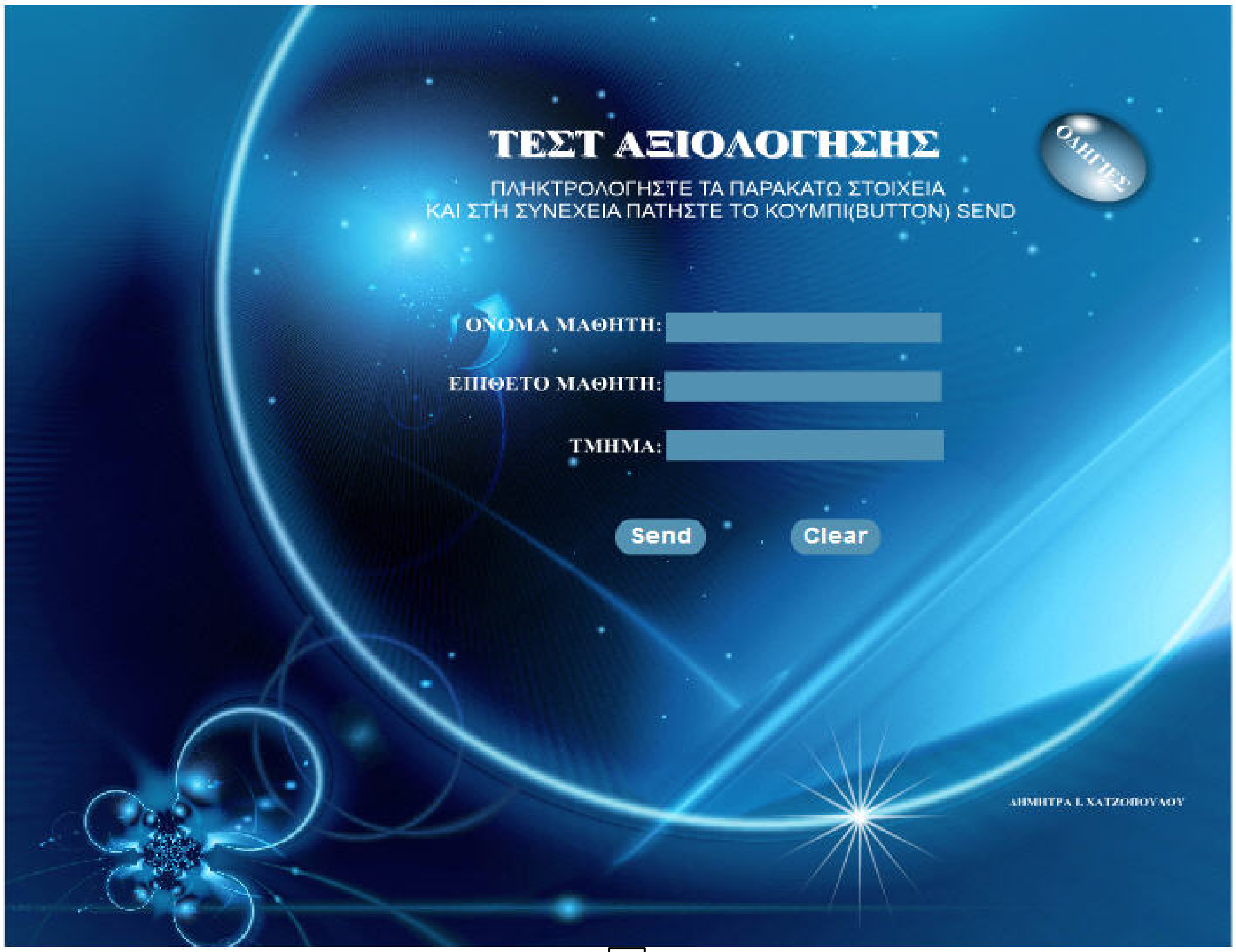
ΔΗΜΗΤΡΑ Ι. ΧΑΤΖΟΠΟΥΛΟΥ
ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ ΜΕΤΑΠΤΥΧΙΑΚΟΥ
ΣΤΑ
ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
2009

B2) MAIN MENU


ΤΕΣΤ ΑΞΙΟΛΟΓΗΣΗΣ
ΕΚΠΑΙΔΕΥΤΙΚΟΣ
ΕΠΙΚΟΙΝΩΝΙΑ
BACKGROUND
ΔΗΜΗΤΡΑ Ι. ΧΑΤΖΟΠΟΥΛΟΥ

B3) CHOICE “ΤΕΣΤ ΑΞΙΟΛΟΓΗΣΗΣ”

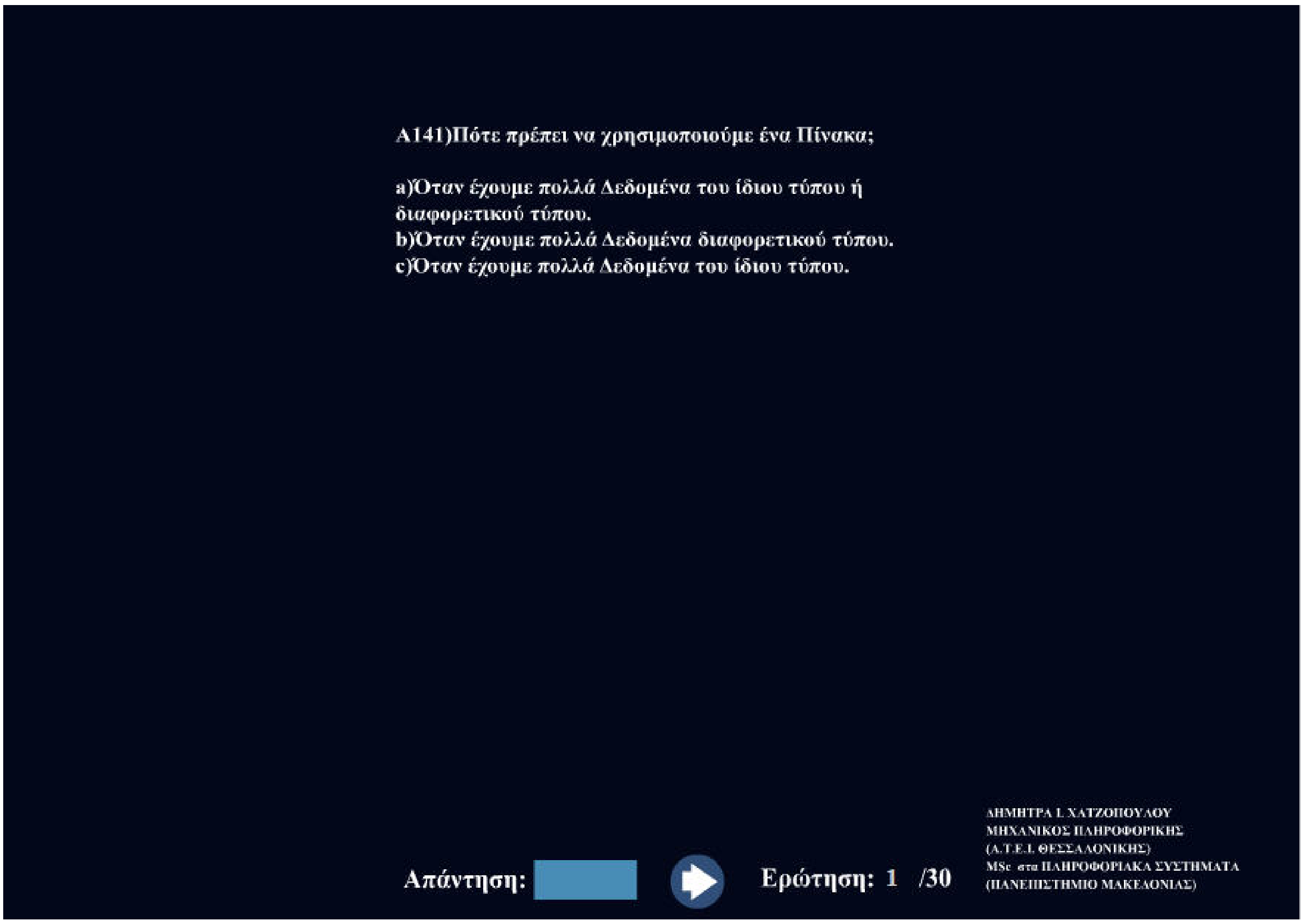
B3.1) STUDENT DETAILS


ΤΕΣΤ ΑΞΙΟΛΟΓΗΣΗΣ

ΠΛΗΚΤΡΟΛΟΓΗΣΤΕ ΤΑ ΠΑΡΑΚΑΤΩ ΣΤΟΙΧΕΙΑ
ΚΑΙ ΣΤΗ ΣΥΝΕΧΕΙΑ ΠΑΤΗΣΤΕ ΤΟ ΚΟΥΜΠΙ(BUTTON) SEND

ΟΝΟΜΑ ΜΑΘΗΤΗ:	
ΕΠΙΘΕΤΟ ΜΑΘΗΤΗ:	
ΤΜΗΜΑ:	

B3.2) P.A.T.

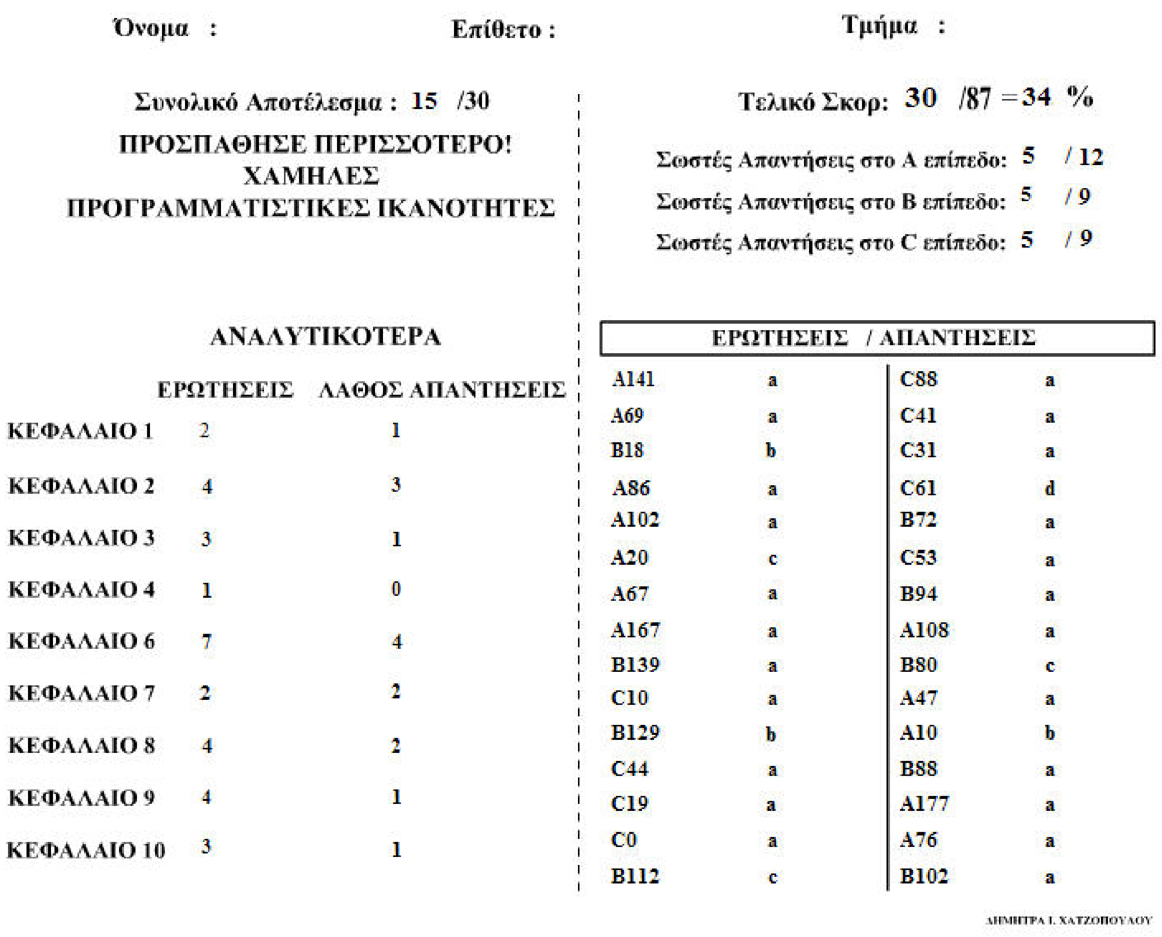

Α141)Πότε πρέπει να χρησιμοποιούμε ένα Πίνακα;

a)Όταν έχουμε πολλά Δεδομένα του ίδιου τύπου ή διαφορετικού τύπου.
b)Όταν έχουμε πολλά Δεδομένα διαφορετικού τύπου.
c)Όταν έχουμε πολλά Δεδομένα του ίδιου τύπου.

Απάντηση: Ερώτηση: 1 /30

ΔΗΜΗΤΡΑ I. ΧΑΤΖΟΠΟΥΛΟΥ
ΜΗΧΑΝΙΚΟΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
(Α.Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ)
MSc στα ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ
(ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ)

B3.3) RESULT TEMPLATE



Όνομα : Επίθετο : Τμήμα :

Συνολικό Αποτέλεσμα : 15 /30

ΠΡΟΣΠΑΘΗΣΕ ΠΕΡΙΣΣΟΤΕΡΟ!
ΧΑΜΗΛΕΣ
ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΕΣ ΙΚΑΝΟΤΗΤΕΣ

Τελικό Σκορ: 30 /87 =34 %

Σωστές Απαντήσεις στο A επίπεδο: 5 / 12
Σωστές Απαντήσεις στο B επίπεδο: 5 / 9
Σωστές Απαντήσεις στο C επίπεδο: 5 / 9

ΑΝΑΛΥΤΙΚΟΤΕΡΑ

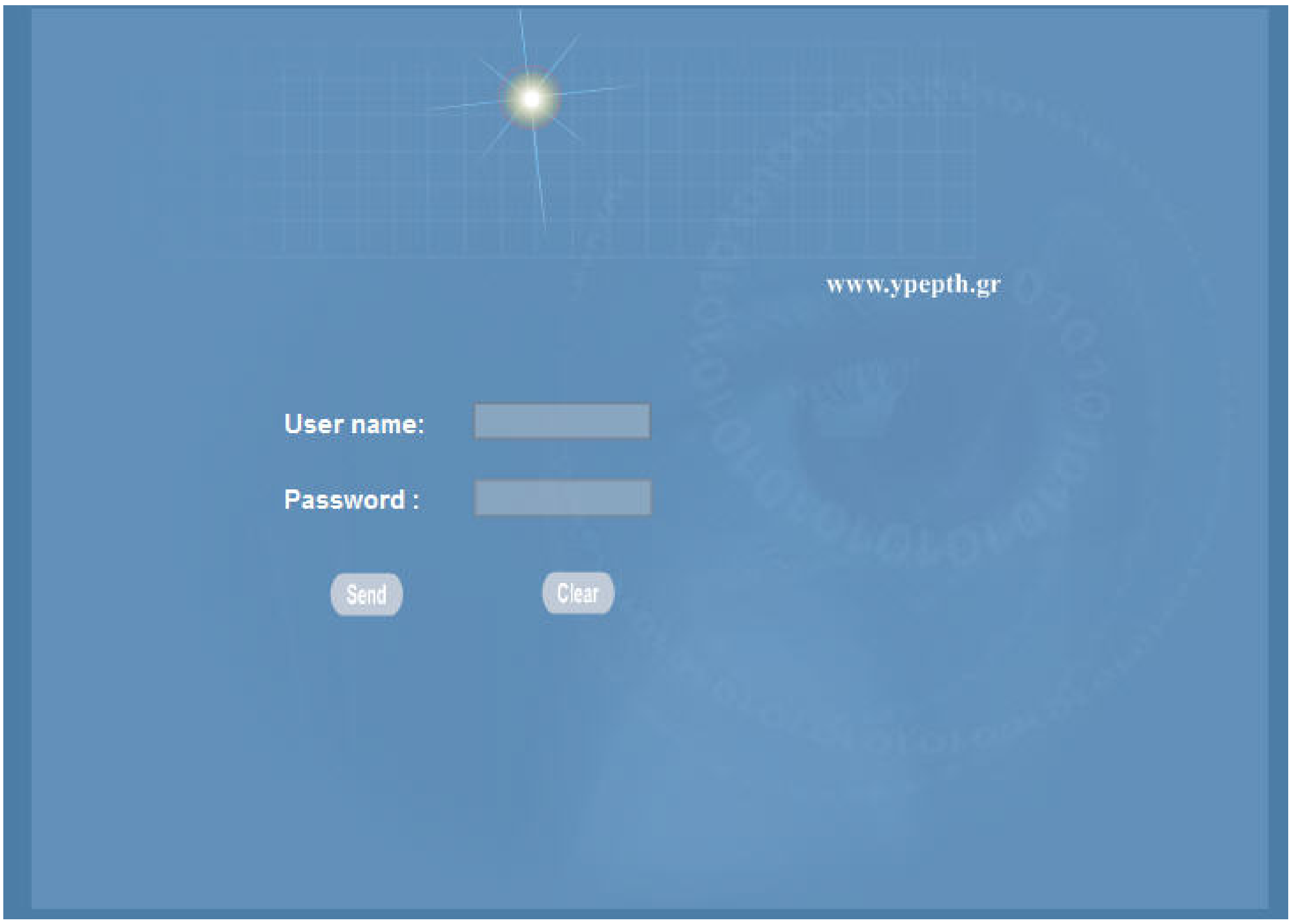
	ΕΡΩΤΗΣΕΙΣ	ΛΑΘΟΣ ΑΠΑΝΤΗΣΕΙΣ
ΚΕΦΑΛΑΙΟ 1	2	1
ΚΕΦΑΛΑΙΟ 2	4	3
ΚΕΦΑΛΑΙΟ 3	3	1
ΚΕΦΑΛΑΙΟ 4	1	0
ΚΕΦΑΛΑΙΟ 6	7	4
ΚΕΦΑΛΑΙΟ 7	2	2
ΚΕΦΑΛΑΙΟ 8	4	2
ΚΕΦΑΛΑΙΟ 9	4	1
ΚΕΦΑΛΑΙΟ 10	3	1

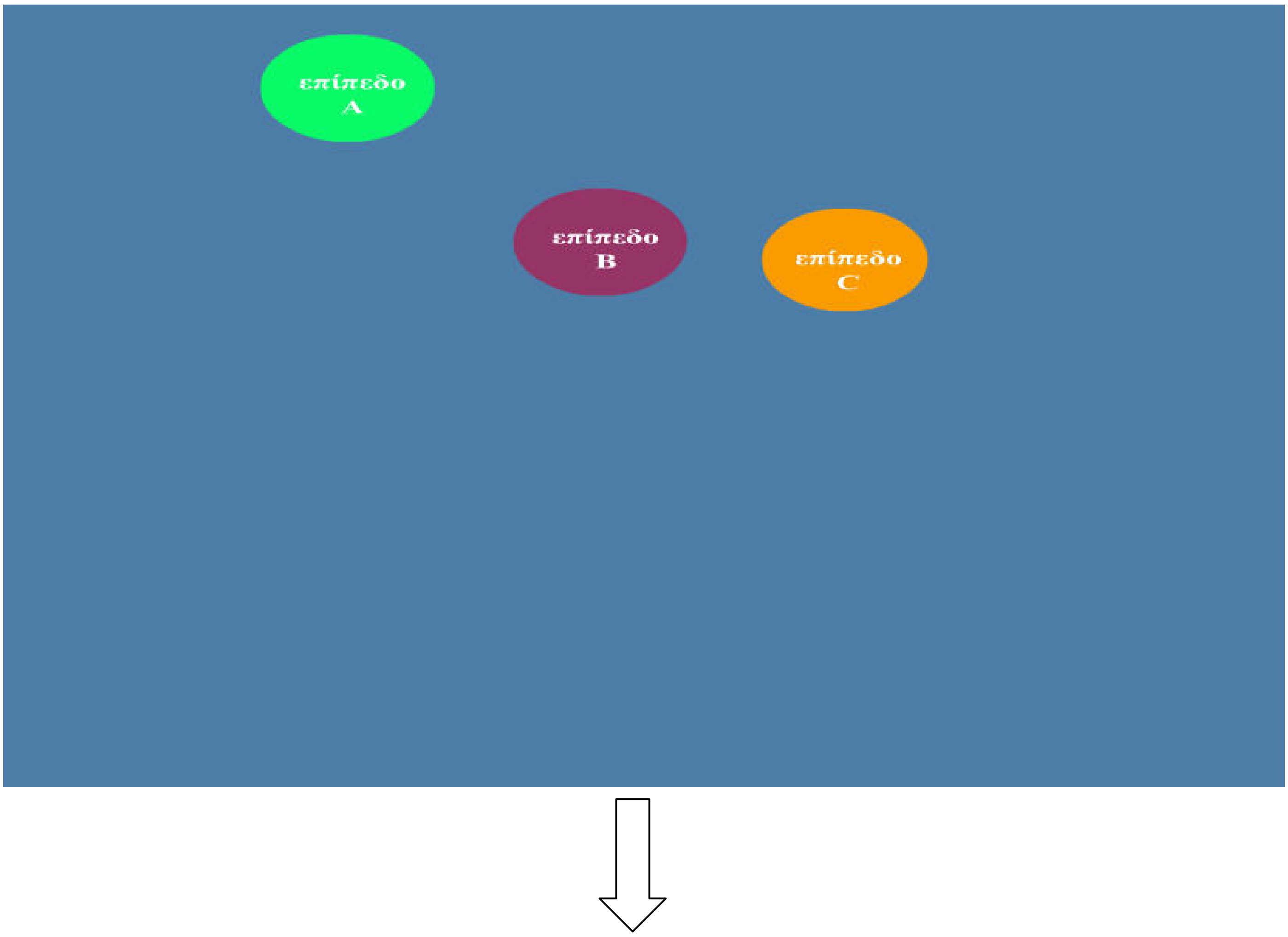
ΕΡΩΤΗΣΕΙΣ / ΑΠΑΝΤΗΣΕΙΣ

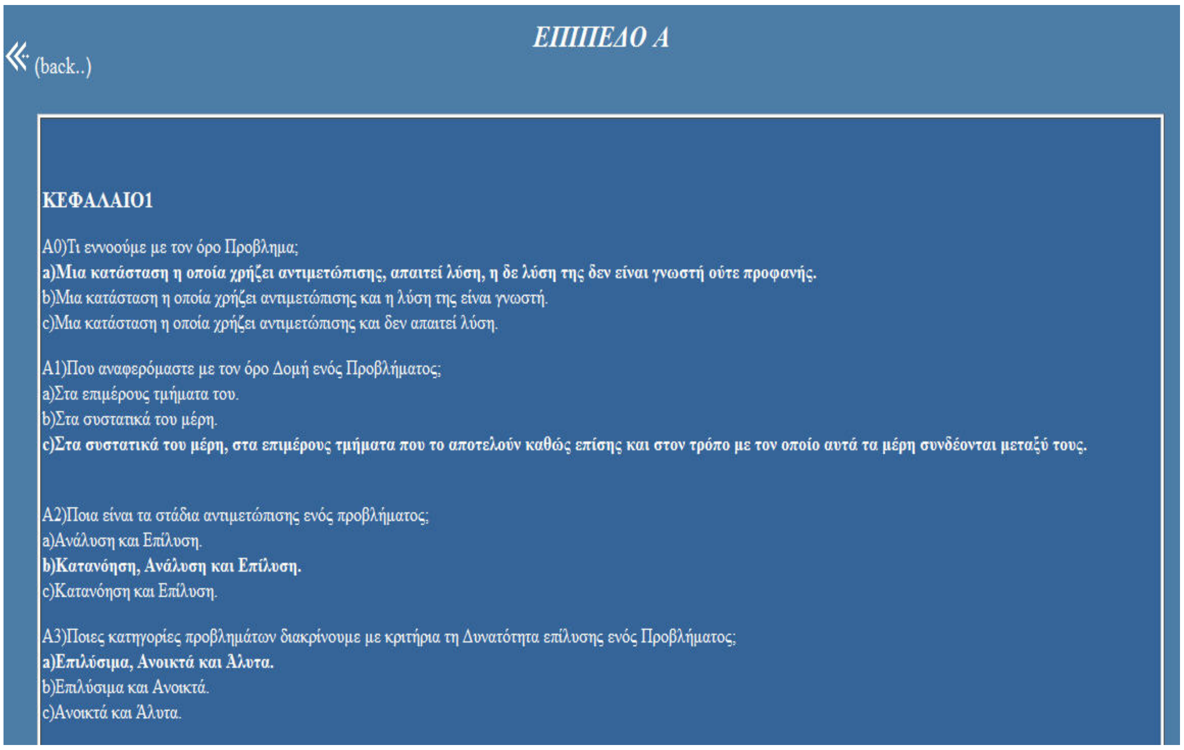
A141	a	C88	a
A69	a	C41	a
B18	b	C31	a
A86	a	C61	d
A102	a	B72	a
A20	c	C53	a
A67	a	B94	a
A167	a	A108	a
B139	a	B80	c
C10	a	A47	a
B129	b	A10	b
C44	a	B88	a
C19	a	A177	a
C0	a	A76	a
B112	c	B102	a

ΔΗΜΗΤΡΑ Ι. ΧΑΤΖΟΠΟΥΛΟΥ

B4) CHOICE "ΕΚΠΑΙΔΕΥΤΙΚΟΣ"




επίπεδο A
επίπεδο B
επίπεδο C



ΕΠΙΠΕΔΟ Α

ΚΕΦΑΛΑΙΟ1

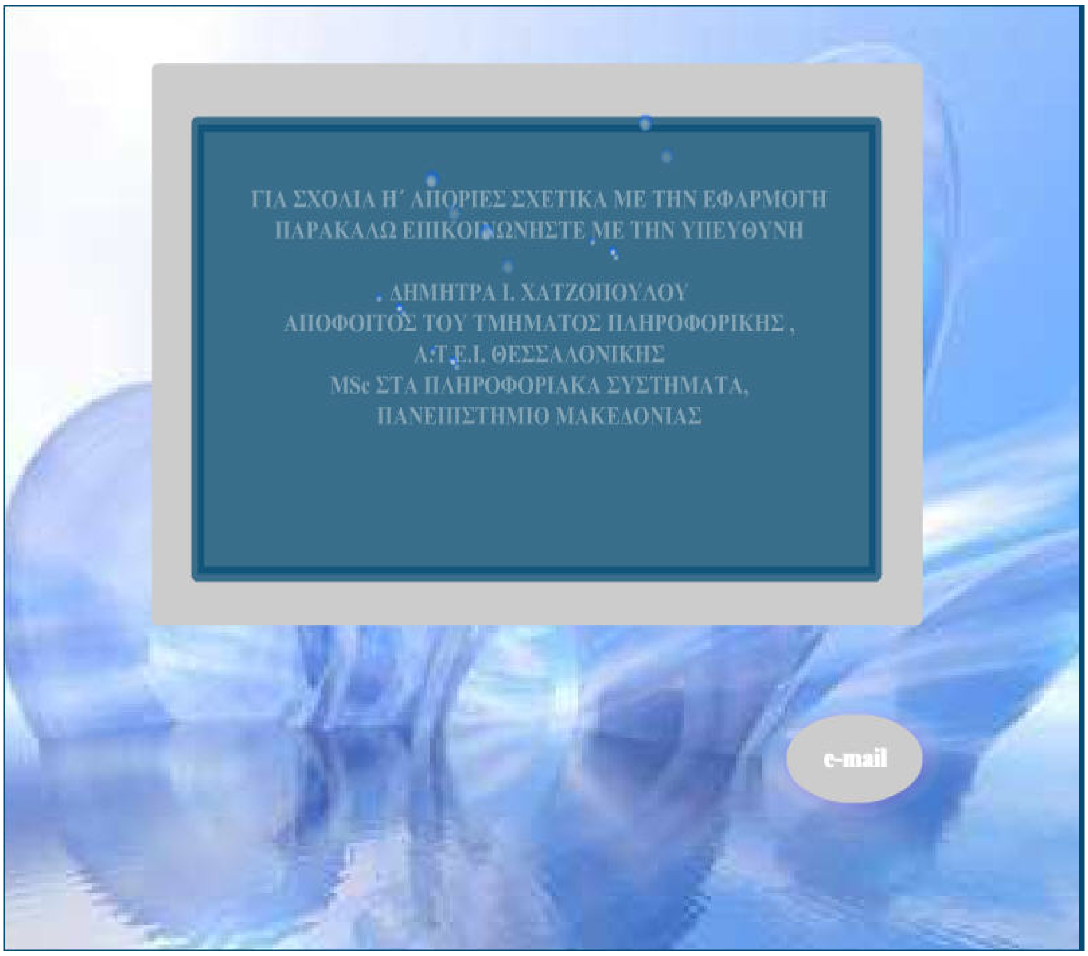
Α0)Τι εννοούμε με τον όρο Προβλημα;
a)Μια κατάσταση η οποία χρήζει αντιμετώπισης, απαιτεί λύση, η δε λύση της δεν είναι γνωστή ούτε προφανής.
b)Μια κατάσταση η οποία χρήζει αντιμετώπισης και η λύση της είναι γνωστή.
c)Μια κατάσταση η οποία χρήζει αντιμετώπισης και δεν απαιτεί λύση.

Α1)Που αναφερόμαστε με τον όρο Δομή ενός Προβλήματος;
a)Στα επιμέρους τμήματα του.
b)Στα συστατικά του μέρη.
c)Στα συστατικά του μέρη, στα επιμέρους τμήματα που το αποτελούν καθώς επίσης και στον τρόπο με τον οποίο αυτά τα μέρη συνδέονται μεταξύ τους.

Α2)Ποια είναι τα στάδια αντιμετώπισης ενός προβλήματος;
a)Ανάλυση και Επίλυση.
b)Κατανόηση, Ανάλυση και Επίλυση.
c)Κατανόηση και Επίλυση.

Α3)Ποιες κατηγορίες προβλημάτων διακρίνουμε με κριτήρια τη Δυνατότητα επίλυσης ενός Προβλήματος;
a)Επιλύσιμα, Ανοικτά και Άλυτα.
b)Επιλύσιμα και Ανοικτά.
c)Ανοικτά και Άλυτα.

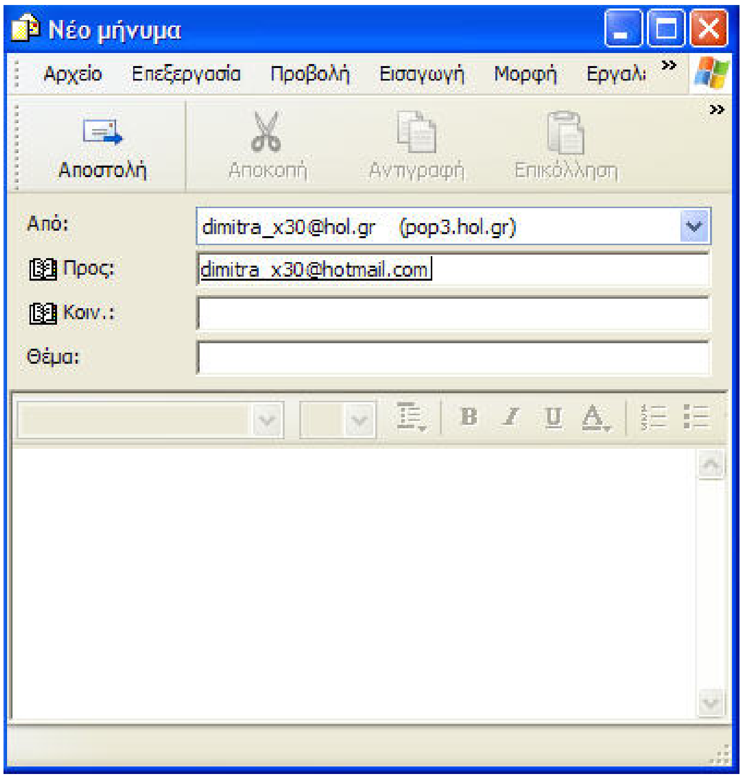
B4) CHOICE “ΕΠΙΚΟΙΝΩΝΙΑ”



ΓΙΑ ΣΧΟΛΙΑ Η΄ ΑΠΟΡΙΕΣ ΣΧΕΤΙΚΑ ΜΕ ΤΗΝ ΕΦΑΡΜΟΓΗ
ΠΑΡΑΚΑΛΩ ΕΠΙΚΟΙΝΩΝΗΣΤΕ ΜΕ ΤΗΝ ΥΠΕΥΘΥΝΗ

ΔΗΜΗΤΡΑ Ι. ΧΑΤΖΟΠΟΥΛΟΥ
ΑΠΟΦΟΙΤΟΣ ΤΟΥ ΤΜΗΜΑΤΟΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ,
Α.Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
MSc ΣΤΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ,
ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

$\Downarrow$



Από: dimitra_x30@hol.gr (pop3.hol.gr)
Προς: dimitra_x30@hotmail.com
Κοιν.:
Θέμα:

REFERENCES

Ahoniemi, T. and Reinikainen, T. (2006). ALOHA – A grading tool for semi – automatic assessment of mass programming courses. Proceeding, Koli Calling2006, pp. 139-140.

Ahoniemi, T., Lahtinen, E. and Reinikainen, T. (2008). Improving pedagogical feedback and objective grading. 39th Symposium on Computer Science Education, Special Interest Group on Computer Science Education'08, March 12-15, 2008, Portland, Oregon, USA, pp.72-76.

Ala-Mutka, K.M. (2005). A survey of automated assessment approaches for programming assignments. Vol. 15, No. 2, June 2005, pp.83-102.

Arnow, D. and Barshay, O. (1999). On-line programming examinations using Web to teach. Innovation and Technology in Computer Science Education ´99, 6/99 Cracow, Poland, pp. 21-24.

Bakali, A., Giannopoulos, I., Ioannidis, N., Kilias, C., Malamas, K., Manolopoulos, J. and Politis, P. (2004). Application development in programming environment, third class in General senior high school of Greece. Organization of school books publications, Athens. Edition 5th , pp. 1-35, 51-66, 79-83, 115-216.

Becker, K. (2003). Grading programming assignment using rubrics. Innovation and Technology in Computer Science Education'03, Proceeding of the 8th annual conference of Innovation and technology in Computer science education, New York, USA, 2003, Association for Computer Machinery Press, pp. 253

Berry, R.E. and Meekings, B.A.E. (1985). A style analysis of C programs. Communications of Association for Computer Machinery,28(1), pp. 80-88.

Bloom, B.S. (1956). Taxonomy of educational objectives. Handbook I. Cognitive Domain, Longmans, Green and Company, pp. 201-207.

Brusilovsky, P. and Sosnovsky, S. (2005). Engaging students to work with self-assessment questions: A study of two approaches. Innovation and Technology in Computer Science Education ´05, June 27-29, Monte de Caparica, Portugal, pp. 251-255.

Califf, M.E. and Goodwin, M. (2002). Testing skills and knowledge: Introducing a laboratory exam in CS1. 33th Symposium on Computer Science Education, Special Interest Group on Computer Science Education ´02,February 27-March 3, Covington, Kentucky, USA, pp. 217-221.

Carter, J., Ala-Mutka, K., Fuller, U., Dick, M., English J., Fone, W. & Sheard, J. (2003). How shall we assess this?. Working Group Reports in Innovation and Technology in Computer Science Education '03, Association for Computer Machinery Press, New York, NY, USA, pp. 107-123.

Choy, M., Lam, S., Poon, C.K., Wang, F.L., Yu, Y.T. and Yuen, L. (2008). Design and implementation of an automated system for assessment of computer programming assignments, The 6th International Conference on Web-based Learning(2007), (LNCS 4823), pp. 584-596.

Daly, C. and Horgan, J. (2001). Automatic plagiarism detection. proceedings of the International Conference in Applied Informatics, pp. 255-259.

Daly, C. and Waldron, J. (2004). Assessing the assessment of programming ability. 35th Symposium on Computer Science Education, Special Interest Group on Computer Science Education´04,March 3-7, Norfolk, Virgina, USA, pp. 210-213.

Denenberg, A.S. (1981). Test construction and administration strategies for large introductory courses. Special Interest Group on Computer Science Education Bulletin, v.13 n.1, Association for Computing Machinery (1981 ACM 0-89791-036-2/81/0200/0235), pp. 235-243.

English, J. (2002). Experience with a computer-assisted frmal programming examination. Innovation and Technology in Computer Science Education ´02, June 24-26, Aarhus, Denmark, pp. 51-54.

Habeshaw, S., Gibbs, G. and Habeshaw, T. (1992). 53 Problems with large classes. Technical and Educational Services Ltd., Bristol, U.K.

Hwang, W-Y., Wang, C-Y., Hwang, G-J., Huang, Y-M. and Huang, S. (2008). A web-based programming learning environment to support cognitive development. Interacting with Computers, Volume 20, Issue 6, December 2008, p.p. 524-534.

Jackson, D. (2000). A semi-automated approach to online assessment, Dept. of Computer Science. Innovation and Technology in Computer Science Education 2000 7/00 Helinski, Finland (ACM2000), pp. 164-168.

Jackson, D. and Usher, M. (1997). Grading student programs using ASSYST. Special Interest Group on Computer Science Education Bulletin , 29(1), pp.335-339.

Jones, A. (1997). Setting objective tests. Journal of Geography in Higher Education, 21(1), pp. 104-106.

Joy, M., Griffiths, N. and Boyatt, R. The BOSS Online Submission and Assessment System. (2005). Association for Computer Machinery Journal on Educational Resources in Computing, Vol.5, No 3, September 2005. Article 2, pp. 1- 28.

Khamis, N., Idris, S., Ahmad, R. and Idris, N. (2008). Assessing object-oriented programming skills in the core education of computer science and information technology: Introducing new possible approach. WSEAS Transactions on Computers, Issue 9, Volume 7, September 2008, pp.

1427-1436.

Kolstad, R.K.. (1994). Applications of conventional and non-restrictive multiple-choice examination items. Clearing House, 67(6), pp. 315-317.

Lister, R. (2001). Objectives and objective assessment in CS1. Thirdy-second Special Interest Group on Computer Science Education Technical Symposium on Computer Science Education (ACM2001), pp. 292-296.

Lister, R. and Leaney, J. (2003a). First year programming let all the flowers bloom. Fifth Australians Computing Education Conference (ACE2003), pp. 221-229.

Lister, R. and Leaney, J. (2003b). Indroductory programming criterion-referencing, and Bloom. 34th Technical Symposium on Computer Science Education SIGCSE '03 (ACM2003), pp. 143-147.

Lister, R. (2004). Teaching Java first experiments with a pigs-early pedagogy. Sixth Australian Computing Education Conference (ACE 2004), Conference in Research and Practice in Information Technology, pp. 177-183.

Lister, R. (2005). One small step toward a culture of peer review and multi-institutional sharing of educational resources: A Multiple Choice exam for first semester students. Australian Computing Education Conference 2005, pp. 155-164.

Mason, D.V. and Woit, D. (1998). Integrating technology into computer science examinations. 29th technical symposium on Computer Science Education Special Interest Group on Computer Science Education 98, in SIGCSE Bulletin, 30(1), pp. 140-144.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., Laxer, C., Thomas, L., Utting, I. and Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. Special Interest Group on Computer Science Education Bull., 33(4), pp.125-180.

Oliver, D. and Dobele, T. (2007). First year courses in IT: A Bloom Rating. Journal of Information Technology Education, Vol.6, pp. 347-359.

Reek, K. (1989). The TRY system-or-how to avoid testing student programs. Special Interest Group on Computer Science Education Bulletin, 21(1), pp. 112-116.

Rhodes, A., Bower, A. and Bancroft, P. (2004). Managing large class assessment. Sixth Australian Computing Education Conference (ACE2004), pp. 285-289.

Schwieren, J., Vossen, G. and Westerkamp, P. (2006). Using software testing techniques for efficient handling of programming exercises in an e-Learning platform. The Electronic Journal of e-Learning, Volume 4, Issue 1, pp. 87-94.

Scott, T. (2003). Bloom's Taxonomy applied to testing in computer science classes. Consortium for Computer Science in Colleges, pp. 267-271.

Spacco, J., Hovemeyer, D., Pugh, W., Emad, F., Hollingsworth, J.K. and Padua-Perez, N. (2006). Experiences with Marmoset: Designing and using an advanced submission and testing system for programming courses. Innovation and Technology in Computer Science Education ´06, June 26-28, Bologna, Italy, pp. 13-17.

Starr, C.W., Manaris B. and Stavley R.H. (2008). Bloom’s Taxonomy revisited: Specifying assessable Learning Objectives in Computer Science. 35th Symposium on Computer Science Education Special Interest Group on Computer Science Education’08 March 12-15, 2008, Portland, Oregon, USA, p.p. 261-265.

Suleman, H. (2008). Automatic marking with Sakai. South African Institute of Computer Scientists and Information Technologists 2008, 6-8 October 2008, Wilderness Beach Hotel, Wilderness, South Africa, p.p. 229-236.

Tanaka-Ishii, K., Kakehi, K. and Takeichi, M. (2004). A Web-based report system for programming course – automated verification and enhanced feedback. Innovation and Technology in Computer Science Education’04, June 28-30, 2004, Leeds, United Kingdom, p.p. 278-285.

Thomson, E., Luxton-Reilly, A., Whalley, J.L., Hu, M., Robbins, P. (2008). Bloom’s Taxonomy for CS assessment. Proc. 10th Australian Computing Education Conference (ACE2008), Wollongong, Australia, pp. 155-161.

Traynor, D. and Gibson, J.P. (2005). Synthesis and analysis of automatic assessment methods in CS1. 35th technical symposium on Computer Science Education SIGCSE ´05, February 23-27, St. Louis, Missouri, USA, pp. 495-499.

Traynor, D., Bergin, S. and Gibson, J.P. (2006). Automated assessment in CS1. Eighth Australian Computing Education Conference(ACE2006), pp. 1-6.

Tremblay, G., Guerin, A., Pons, A. And Salah, A.(2008). Oto, a generic and extensible tool for marking programming assignments. Software:Practice and Experience, Vol. 38, No. 3, p.p. 307-333

Wang, F.L. and Wong T.L. (2008). Designing programming exercises with computer assisted instructions. International Conference on Hybrid Learning 2008, LNCS 5169, pp. 283-293.

Whalley, J.L., Lister, R., Thompson, E., Clear, T., Robbins, P., Kumar, P.K.A. and Prasad, C. (2006). An Australasian study of reading and comprehension skills in novice programmers, using Bloom and Solo Taxonomies. 8th Australian Computing Education Conference, Hobart, Australia, p.p. 243-252.

Woit, D. and Mason, D. (2003). Effectiveness of Online Assessment. 34th

Symposium on Computer Science Education SIGCSE´03, February 19-23, Reno, Nevada, USA, pp. 137-141.

Wilson, T.L. and Coyle, L. (1991). Improving multiple choice questioning: Preparing students for standardized test. Clearing House, 64(6), pp. 421-423.

Yu, Y.T., Poon, C.K. and Choy, M. (2006). Experiences with PASS: Developing and using a programming assignment assessment system. Proceedings of the Sixth International Conference on Quality Software (QSIC'06), p.p. 360-368.