A System for Assessing the Knowledge and Skills of Students in Computer Programming

Nikolay Kirov
New Bulgarian University
ul. Montevideo 21, Sofia, Bulgaria
Email: nkirov@nbu.bg

***Abstract*—This article presents the author's experience in teaching programming to university students in first and second year. The system for assessing the knowledge and skills of students is an essential part of teaching. It aims not only to assess students, but also to help improving their knowledge. The importance and difficulty of Computer Programming requires specific and unconventional approach to this activity. The article discusses all three elements of ongoing assessment (tests for knowledge, homeworks, and practical programming skills), as well as the rules for final exams and final assessment.**

I. Introduction

"Assessment is an ongoing process aimed at understanding and improving student learning. It involves making expectations explicit and public; setting appropriate criteria and high standards for learning quality; systematically gathering, analyzing, and interpreting evidence to determine how well performance matches those expectations and standards, and using the resulting information to document, explain, and improve performance." Angelo [12].

Teaching of computer programming for undergraduate students is a difficult task for every teacher. It is determined of specific features of programming.

This subject is related to the acquisition of practical skills in writing programs and the successful learning concludes with unaided writing of program codes. For that reason teaching programming is composed in two parts – lectures and labs.

In this paper we have in mind three courses which usually are proposed from first to third semester: Introduction to programming, Object-oriented programming, and Algorithms and data structures

II. Points assessment system

Assessment rules of knowledge and skills are announced at the beginning of every semester. They are also published in the course website.

The assessment system is based on points. The student collects points during the semester and the final grade depends of these points. The next left side table shows the correspondence of the obtained points and the grades in the scale of grades based on six. The right side table represent the US-grade system, applied for a programming course [11]:

Points	Mark
0 – 49	2
50 – 59	3
60 – 75	4
76 – 89	5
90 – 100	6

Total	Grade
≥ 93	A
90 to < 93	A–
87 to < 90	B+
83 to < 87	B
80 to < 83	B–
75 to < 80	C+
70 to < 75	C
65 to < 70	C–
63 to < 65	D+
60 to < 63	D
50 to < 60	D–
0 to < 50	F

A. Current Assessment and Semester Exam

Universities' assessment systems in England and the U.S. allow the student to obtain final evaluation in an academic discipline at the end of the semester without having to appear on semester examination. For this purpose during the semester students are gaining points from the performance of different tasks. In some other assessment systems, where the semester exam is mandatory, the current assessment gives only a part of the final grade.

The current assessment are formed of:

- Three tests – first, second and final ($10+10+20 = 40$ points)
- Three exams of practice ($10 + 10 + 10 = 30$ points)
- Three homeworks ($10 + 10 + 10 = 30$ points)

Usually the lecturer/instructor gives additional "bonus" points at the end of the semester. They are not more than 10 and may be given for regular attendance of classes, participation and performance in programming competitions, for answers to questions asked during lectures, etc.

In two cases, the student must appear at the semester (terminal) exam: if he/she has collected under 50 points in the current assessment or if he/she wants to raise his/her final score.

Semester examination consists of:

- Final test for 40 points.
- Final control practice for 20 points.
- Written exam for 20 points.
- Interview (oral exam) for 20 points.

Usually the last lecture of the semester is the final test, and the labs are final exam of practice. The results of these are counted for formation of current (continuous) assessment as well as for semester exam if the student should pass such examination. On the date of the semester exam, students can optionally to fill the final test and/or final exam of practice. In this case if he/she gains more points we count these points for final evaluation.

III. Tests

The students' tests are of multiple choice type. Each question has proposals for 4 answers marked with a), b), c) and d). The proposals can be correct or incorrect (true or false). For each question all variations (with repetitions of elements true and false) are possible, i.e.

- All are correct `tttt`,
- Three are correct `tttf, ttft, tftt, fttt`,
- Two are true `ttff, tftf, tfft, fttf, ftft, ffft`,
- One is correct `tfff, ftff, fftf, ffft`,
- No correct `ffff`.

The number of options is $4^2 = 16$, so the probability of guessing the answer is 6.25%.

In the test class each student receives a piece of paper with the test. All tests are different, which largely eliminates the element of copying from the neighbour in completing the test.

A. Preparation

Preparation of the test involves the choice and formulation of the questions and selecting answers (proposals for answering) – true and false. Since each question in an individual test contains four proposals, the question should have at least 4 responses. In order to avoid copying from student's neighbor (in the case of identical responses), it is a good idea to prepare at least 12 suggestions for answering each question. The questions and proposals are prepared in the form of a plain text file. This file is input data to a computer program that generates individual tests. We use a random number generator to select the questions and the answers for each individual test. Also a 4-digit identifier is generated for each test. Most often an individual test contains from 10 to 20 questions. Optionally, we can give the total number of correct answers.

The program generates a LaTeX [5] file with the specified number of tests. After processing the file, we obtain `ps` or `pdf` file which is ready for printing. The program also generates a file with the answers of every individual test.

At least one week before the test day the questions and two sample answers (correct and incorrect propositions) are published on the course website. Thus, students have the opportunity to learn pre-test and can contact the instructor for clarification or consultation.

B. Implementation

On the test day, prior to the distribution of individual tests the rules are recalled:

- During the test students can use lectures, textbooks and any other printed materials.
- On a separate sheet of paper or using a special template students must write their answers:
  - positive (yes, true, correct),
  - negative (no, false, incorrect) or
  - neutral (–, nothing, I don't know).
- If the students have questions about ambiguities in the test during the test time, the instructor can answer the questions personally.

When a question was asked by several students it is announced to all, along with the answer. Sometimes lecture slides can be shown which explicitly or more often implicitly contains the answer of the question.

Now about the phenomenon of copying and cheating. The classical approach is that when the student make a test or quiz, he/she must show what has been learned and hence it is prohibited for him/her to use external sources of information. In this case the learning material should be memorized and then remembered or reproduced or should be applied – for example in solving problems or writing programming code. In the specific activity of programming, the main goal is the skill to writing correct code, i.e. the final product is a computer program. This is a largely creative process and limiting use of information is completely pointless. Suppose, for example, a manager of some IT company gives the programmer a task to write a software and not allow him to use literature. On the contrary, the requirement is usually to find useful manuals and books and to create software according to the latest trends in the specific areas. I think this approach should be applied at the "student bench", too.

The purpose of the test is not only to check students' knowledge, but also urge them to open the textbook, to seek specific information there and to make logical connections between the test questions and the text written in the textbook.

The time for a test is two academic hours which is equal to one lecture in many universities.

C. Checking by the teacher

In checking of each individual test, every question receives test points in the range $[-4, 4]$. They are the sum of points for each answer, which can be:

- $+1$ when the student gave the correct answer (yes or no);
- -1 for an incorrect answer;
- 0 if answered "do not know".

For entire individual test the points for each question is summarized. Let the test contain N questions, and let one particular test collect t test points ($t \leq 4N$). If the test gives M exam points, this individual test gives the student $e = \dfrac{tM}{4N}$

exam points. If e is not an integer, we consider the smallest integer greater than e – rounding up, i.e. in favor of student.

Often in the literature we can find recommendations of the type: "It is not recommended assignment of penalty points for an incorrect answer because then the tested student is afraid to answer, if not fully satisfied, and so he/she does not show true level of his/her knowledge/skills"[3]. My argument for not complying with this rule is again that the programming is a specific activity and knowledge in this area must be clear – the programmer has to give a clear account of what is known and what is not known. One tiny mistake in a computer program (e.g. absence of a comma) can lead to completely unpredictable consequences – from "innocent" spelling error in some text to endangering human life or loss of spacecraft.

D. Checking by the student

At the beginning of next lecture the lecturer announces test results and return tests to the students. Each student should carefully see what is wrong, to consider whether he/she agrees with the marked mistakes and if something is not clear, ask the instructor. The goal of this check-up is that the students realize their mistakes, which obviously helps for better acquisition of the educational material.

Furthermore, it is not easy to formulate short and exact questions. Also some proposals for answers can be not very clear. A good practice is to accept the views of students who interpret some questions or responses differently and to increase accordingly their exam points.

IV. HOMEWORK

There are three homeworks in one semester. Students have between one and two weeks from setting the homework to its submission for check-up.

First, a list of tasks is created, depending on the number of students and nature of the material. Some tasks are directly related to the exercises in the textbook, while others are extensions of the examples discussed in the lectures and there are such for which the student must find and study topics on the Internet (for example algorithms).

The fixing of a task number to a particular student depends on his/her faculty number. If there are n prepared tasks and faculty number of the student is F, then the number of his/her task is remainder of integer division (arithmetic operation C and C++): `f % n` or math module: f mod n. Thus, normally one task falls on more than one student. This stimulates teamwork for solving task, which is an important element of the preparation of programmers. Indeed, the possibility remains one student to do the homework, and others to copy it.

Completed programs in form of source code are sent to the instructors of labs via e-mail or using the Learning Management System "Moodle". The check-up and evaluation is done in the presence of students. Sometimes, but for some courses as a rule, leading the exercises requires the student himself/herself to submit the homework and bring the text of the program, compile it, test the code with the examples set by the instructor and explain the algorithms and their implementations. Thus, the assessment may be reduced if it is apparent that the student has copied the program and not able to clarify it.

V. EXAM OF PRACTICE

Exams of practice are made in lab classes. They are assignments for writing programming codes. The codes should be able to compile successfully and then to pass the tests with inputs given by the instructor. Also, the students should be able to explain the algorithms and techniques used in writing the codes. The exams of practice are the most important elements of the evaluation, because they summarize the knowledge from the lectures and the experience from labs classes and homeworks.

VI. WRITTEN EXAMINATION AND DISCUSSION

The written exam is the last phase of the semester examination. Students have already collected the points from final test and final practice (in the range $[10, 60]$). When the points are ≥ 50, the student may refuse the written exam and interview. When the points are 10, the student must earn all 40 points from the written exam and interview to reach a positive grade.

Students draw their exam questions in a classical method. A list of questions may be given in advance or a question may be a topic from a lecture. Another possible option is the first question to be the type: "choose the question you know best" and the second question to be selected by the instructor. This approach eliminates the case: "I did not learn only one question and it fell on me".

The interview is a discussion upon the paper written by the student. It emphasizes written statements that are vague, ambiguous, not entirely true, true with addition or false, trying to get the student to think and to clarify what he/she meant when he/she wrote these statements. If there is a gap in the text the student may be required to add text, or to set an example, or to write small source code. Typically, the interview begins with a small arithmetic – how many points the student has collected till now and what grade can be reached. For example, if he/she is collected 45 exam points, 5 points from this phase mean grade 3, 15 points – grade 4 or 35 points for 5. At this session, the student can not complete for an excellent grade because of weak performance in the test and the final exam of practice.

VII. CONCLUSION

The idea for such an evaluation system occurred in 1999 when I started teaching programming at the South West University, Blagoevgrad. Over the years I changed some details and now the system is applied in New Bulgarian University, Sofia for several courses [10].

An important element in the use of tests is the analysis of the results from a specific test where you can make statistical processes to assess the test questions and answers, and others. It is a good prospect for future work.

REFERENCES

[1] Cay Horstmann, Computing Concepts with C++ Essentials, Third Edition, John Wiley & Sons, 2003.

[2] G. Bizhkov, G. Theory and Methods of teaching tests, “Education”, S. 1992 (in Bulgarian).

[3] Handbook of assessment, Assessment Centre, NBU, 2007 (in Bulgarian).

[4] H. M. Deitel, P. J. Deitel, C++ How to Program, Fourth edition, Prentice Hall, 2002.

[5] LaTeX – A document preparation system.
http://www.latex-project.org/

[6] M. Goodrich, R. Tamassia, D. M. Mount, Data Structures and Algorithms in C++, Wiley, 2004.

[7] N. Kirov, A collection of teaching materials: Introduction to programming Demetra, Sofia, 2003 (in Bulgarian).

[8] N. Kirov, A collection of teaching materials: Programming and Data Structures, Demetra, Sofia, 2004 (in Bulgarian).

[9] N. Kirov, Teaching programming in high schools, Mathematics and Mathematics Education, 2001 (in Bulgarian).

[10] Nikolay Kirov home page – teaching.
http://nikolay.kirov.be/teach_en.html

[11] Sillabus – Data Structure and Algorithm, Department of Computer Science, California State University San Marcos.
http://courses.csusm.edu/cs311ah/Pages/Syllabus.htm

[12] T. Angelo, Reassessing (and defining) assessment. AAHE Bulletin, 48(3), (1996), 7-9.