CSECS 2014, pp. 000 - 000

The 10 Annual International Conference on
Computer Science and Education in Computer Science,
July 03-07 2014, Albena, Bulgaria

A TEST SYSTEM FOR CHECKING AND EVALUATION THE STUDENTS' PROGRAMMING KNOWLEDGE

Nikolay KIROV

Abstract: The article describes the preparation and implementation of multiple-choice type tests for checking the students' knowledge of programming. Justification of the chosen approach, methodology and technical details are discussed. Software for generating tests and analysing of results is proposed.

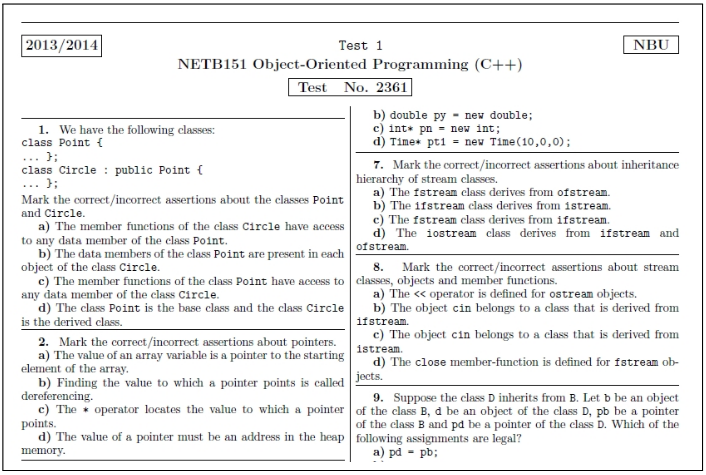
Keywords: *Multiple choice tests, assessment of programming, software.*

ACM Classification Keywords: *K.3.2 [Computers and Education]: Computer and Information Science Education* Computer science education
J.1 [Computer Applications]: Administrative Data Processing Education

Introduction

Different types of tests are used to assess the knowledge and skills of students at all levels of education. The most common are multiple-choice and constructed-response tests (see [2]). Our choice is for multiple-choice questions implemented as paper-and-pencil test. We use the following terminology: The test consists of items. The stem is the introductory question or statement at the beginning of each item. It is followed by options. An option may be an answer – the correct option or a distractos – the incorrect option. The items are stored in an item bank. Items are pulled from the bank and assigned to test forms for publication either as a paper-and-pencil test or some form of e-assessment.

Our system uses all the items from the item bank to produce files for paper-and-pencil test. An individual test (test form) consists of fixed number of items with four options for any item (Figure 1). All individual tests are different because they are produced from the item bank using random number generator. Randomized paper-and-pencil test forms are used to minimize copying among examinees in adjacent seats.



2013/2014 Test 1 NBU

NETB151 Object-Oriented Programming (C++)

Test No. 2361

1. We have the following classes:

```
class Point {
... };
class Circle : public Point {
... };
```

Mark the correct/incorrect assertions about the classes `Point` and `Circle`.

a) The member functions of the class `Circle` have access to any data member of the class `Point`.
b) The data members of the class `Point` are present in each object of the class `Circle`.
c) The member functions of the class `Point` have access to any data member of the class `Circle`.
d) The class `Point` is the base class and the class `Circle` is the derived class.

2. Mark the correct/incorrect assertions about pointers.
a) The value of an array variable is a pointer to the starting element of the array.
b) Finding the value to which a pointer points is called dereferencing.
c) The $*$ operator locates the value to which a pointer points.
d) The value of a pointer must be an address in the heap memory.

b) `double py = new double;`
c) `int* pn = new int;`
d) `Time* pt1 = new Time(10,0,0);`

7. Mark the correct/incorrect assertions about inheritance hierarchy of stream classes.
a) The `fstream` class derives from `ofstream`.
b) The `ifstream` class derives from `istream`.
c) The `fstream` class derives from `ifstream`.
d) The `iostream` class derives from `ifstream` and `ofstream`.

8. Mark the correct/incorrect assertions about stream classes, objects and member functions.
a) The << operator is defined for `ostream` objects.
b) The object `cin` belongs to a class that is derived from `ifstream`.
c) The object `cin` belongs to a class that is derived from `istream`.
d) The `close` member-function is defined for `fstream` objects.

9. Suppose the class `D` inherits from `B`. Let `b` be an object of the class `B`, `d` be an object of the class `D`, `pb` be a pointer of the class `B` and `pd` be a pointer of the class `D`. Which of the following assignments are legal?
a) `pd = pb;`

Figure 1: The upper part of a real individual test

The presented here system produces individual tests of multiple-choice type. The number of correct options (answers) of an item may be 0, 1, 2, 3 or 4, i.e. we have multiple response items. Also a specific requirement is that the student have to identify each option or as answer (using "true" or "yes") either as distractor (using "no" or "false"), i.e. he or she has 3 choices of response:

- "yes", i.e. I know the option is an answer;
- "no", i.e. I know the option is a distractor;
- nothing, i.e. I do not know whether the option is an answer or a distractor.

A correct response ("yes" or "no") wins a point but an incorrect response loses one point (penalty point) in the total score. Some arguments for these non standard characteristics (0, 1 or many answers per item, 3-valued logic and penalties) of our multiple-choice test for students in programming can be found in [1]. About penalties, for example, the SAT test system removes a quarter point from the test taker's score for an incorrect answer.

Advantages and Disadvantages

The advantages of multiple-choice tests are well known (see for example [2]) and will not be discussed here. More important is how to avoid the disadvantages of this type of testing. Our modifications of the standard multiple-choice test and the use in a particular area of knowledge gives us a chance to substantiate our approach. We will discuss three of the most widespread arguments.
1. The most serious disadvantage is the limited types of knowledge that can be assessed by multiple-choice tests. Multiple choice tests are best adapted for testing well-defined or lower-order skills.
Introductory courses on programming possess the characteristics: the ideas, methods and rules in programming are well-defined, and an essential part of programming skills is the low-order knowledge: syntax, simple constructions, etc. Our modifications to the standard multiple-choice test contribute to setting more sophisticated and complex questions that check students' knowledge at a higher level. The test is not alone and is not sufficient for a complete assessment of student programming skills.
2. Another disadvantage of multiple-choice tests is possible ambiguity in the student's interpretation of the item.
We apply several methods in avoiding this disadvantage. At least one week before the date of the test all original stems and two example options per item (an answer and a distractor) are published online on the course website. Thus, the students have the opportunity to learn about the test in advance. In addition a few days before the test, a general advice is organized on which the issues of the test are discussed. During the test time students can use lectures, textbooks and any other printed materials. Sometimes students are allowed to use a computer, compiler, and even the Internet (arguments for this can be found in [1]). If a student has questions about ambiguities in the test stems or options, the instructor answers the questions personally.

After checking the test by the instructor, the test forms are returned to the students at the next class. Each student should carefully check his or her individual test in order to determine whether he or she agrees with the errors noted. If something is not clear she or he discuss the case with the instructor. The goal is the students to realize their mistakes, which obviously contributes to a better understanding of the teaching material. Our practice is to accept the students' opinions when the student has different interpretations and possibly to increase his or her test points.
3. Another disadvantage of multiple-choice examinations is that a student who is incapable of answering a particular question can simply select a random answer and still have a chance of receiving a mark for it.
We assess a test which consists of `M` items and is obtained `x` points as follows:

```
if ( x/M/4*100 >= 90 ) e = 6;
if ( x/M/4*100 >= 76 ) e = 5;
if ( x/M/4*100 >= 60 ) e = 4;
if ( x/M/4*100 >= 50 ) e = 3;
else e = 2;
```

where `e` is the mark in a six point system of marks. To calculate the probability of passing the test using random method, we choose a test with 10 items and maximum 40 points. Probability is 0.11% in the case that the student has marked all options with "yes" or "no" (given the answer to all the questions), and the probability is 0.0034% if the student is checked randomly with "yes", "no" or nothing.

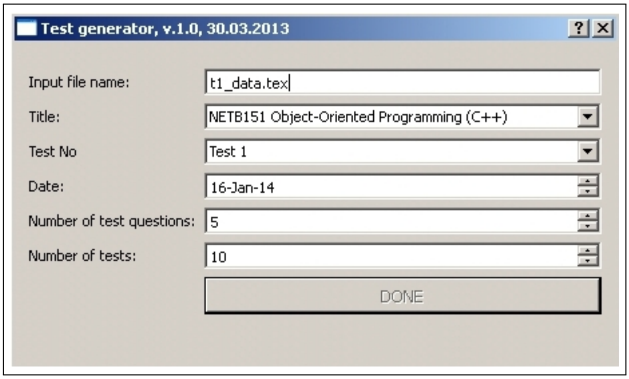
The Software

Preparation of the test begins with selecting items – stems and options and put them into an item bank. At least 10 items, each having at least 5-6 possible answers, should be completed and stored as a plain text file in the following format: The stem text, list of options each on a new line with prefix "+" for answers or "−" for distractors. The example bellow shows a part of an item – the stem following with two answers and one distractor.

```
Mark the correct/incorrect assertions about pointers.
+ A pointer denotes the location of a value in the memory.
+ Finding the value to which a pointer points is called
```

```
dereferencing.
- The value of a pointer must be an address in the heap
memory.
```

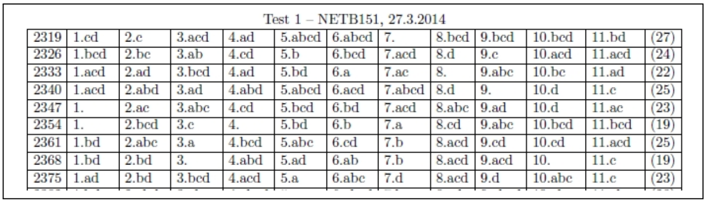
This plain text file is in LaTeX format and represents our item bank. It contains input data for the software tool `test_generator` (Figure 2).


Test generator, v.1.0, 30.03.2013

Input file name: t1_data.tex
Title: NETB151 Object-Oriented Programming (C++)
Test No Test 1
Date: 16-Jan-14
Number of test questions: 5
Number of tests: 10

Figure 2: Test generator – user interface

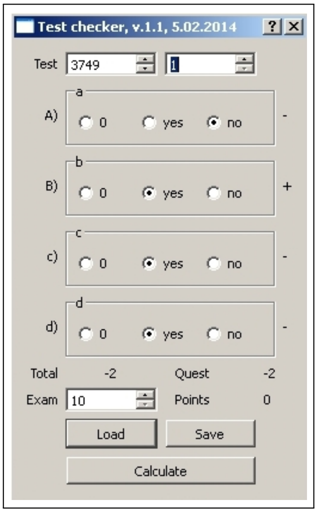
`test_generator` produces individual tests using random distribution of both items and their options. Each individual test consists of 10-20 items with four options, marked as a), b), c), d). The output plain text file (`out.tex`) is in LaTeX format and contains all individual tests (see Figure 1). The second output file (`tab.tex`) is a table for checking the tests (Figure 3). The third output file (`data.txt`) is a copy of the input file with additional data for the generated individual tests.
Checking tests can be carried out manually – using the table (Figure 3) generated by `test_generator` or automatically by the second software tool – `test_checker`.
The input of student responses (completed individual tests) can also be done manually – using the user interface of `test_checker` (Figure 4) or automatically – using a special template for students' answers and scanning the pa-



Test 1 – NETB151, 27.3.2014

2319	1.cd	2.c	3.acd	4.ad	5.abcd	6.abcd	7.	8.bcd	9.bcd	10.bcd	11.bd	(27)
2326	1.bcd	2.bc	3.ab	4.cd	5.b	6.bcd	7.acd	8.d	9.c	10.acd	11.acd	(24)
2333	1.acd	2.ad	3.bcd	4.ad	5.bd	6.a	7.ac	8.	9.abc	10.bc	11.ad	(22)
2340	1.acd	2.abd	3.ad	4.abd	5.abcd	6.acd	7.abcd	8.d	9.	10.d	11.c	(25)
2347	1.	2.ac	3.abc	4.cd	5.bcd	6.bd	7.acd	8.abc	9.ad	10.d	11.ac	(23)
2354	1.	2.bcd	3.c	4.	5.bd	6.b	7.a	8.cd	9.abc	10.bcd	11.bcd	(19)
2361	1.bd	2.abc	3.a	4.bcd	5.abc	6.cd	7.b	8.acd	9.cd	10.cd	11.acd	(25)
2368	1.bd	2.bd	3.	4.abd	5.ad	6.ab	7.b	8.acd	9.acd	10.	11.c	(19)
2375	1.ad	2.bd	3.bcd	4.acd	5.a	6.abc	7.d	8.acd	9.d	10.abc	11.c	(23)

Figure 3: Table for checking the tests manually



Test checker, v.1.1, 5.02.2014

Test 3749 1

A) a: ( ) 0 ( ) yes [✓ no] -

B) b: ( ) 0 [✓ yes] ( ) no +

c) c: ( ) 0 [✓ yes] ( ) no -

d) d: ( ) 0 [✓ yes] ( ) no -

Total -2 Quest -2

Exam 10 Points 0

Figure 4: Test checker – user interface

pers. An input for `test_checker` is also the file `data.txt`, which is produced by `test_generator`. The program `test_checker` creates a text file (`save.txt`), containing audited tests.
Here is a part of the file `save.txt`. The last item (`q11`) of an individual test has 4 correct answers (`no`, `no`, `yes`, `no`) and total 32 points. The next individual test

is number 2347 and the first item (`q1`) gives away -1 points as the student has marked a) and b) as `yes` – incorrect, c) as nothing and d) as `no` – correct.

```
q11
a no +
b no +
C yes +
d no +
4
Total 32 ( 8 pt )
Test No. 2347
q1
a yes -
b yes -
c
d no +
-1
```

After entering all the completed individual tests, `test_checker` gives the results – for each option of each item of the item bank it calculates two sets of numbers. The set $A = \{a, a_1, a_2, a_3\}$ represents all the tests and the set $B = \{b, b_1, b_2, b_3\}$ represents the tests which collect at least a half of maximum points. Here a, b are the number of individual tests which contain the corresponding item and four of its options, a_1, b_1 – the numbers of tests without response (for 0 point), a_2, b_2 – the numbers of tests with the correct response (for 1 point) and a_3, b_3 – the numbers of tests with the incorrect (opposite) response (for -1 points). We have $a_1 + a_2 + a_3 = a$, $b_1 + b_2 + b_3 = b$, $b \leq a$ and $b_i \leq a_i$ for $i = 1, 2, 3$. The file `data_result.txt` contains these numbers as an extension to the any option of the items in the item bank.

```
Mark the correct/incorrect assertions about pointers.
+ A pointer denotes the location of a value in the memory.
% 6:2:3:1  3:0:3:0
+ Finding the value to which a pointer points is called
dereferencing. % 7:2:3:2  3:1:1:1
- The value of a pointer must be an address in the heap
memory. % 6:1:4:1  2:0:1:1
```

The second output file (`data_result1.txt`) is a plain text file containing only the numbers and is suitable for further processing of the results.

Example

The example is the first test of Object-oriented programming (for undergraduate students, second semester) with the following parameters: The item bank consists of 12 item and the number of options for these items are: (14,16,19,16,12,7,12,16, 10,9,16,22). Any individual test consists of 11 items, which means maximum 44 points. `test_checker` generates 30 individual tests but only 19 of them were completed by the students.

The test results are presented on Figure 5. The X-axes presents all items in the item bank, Y-axes – number of tests. The left chart is for all 19 completed individual tests. A little more information about the difficulty of the test items and the students' knowledge is obtained from the right chart, where the test forms of those students who passed the test are counted. These are 11 individual tests, which are collected at least a half (22) of the maximum points (44).

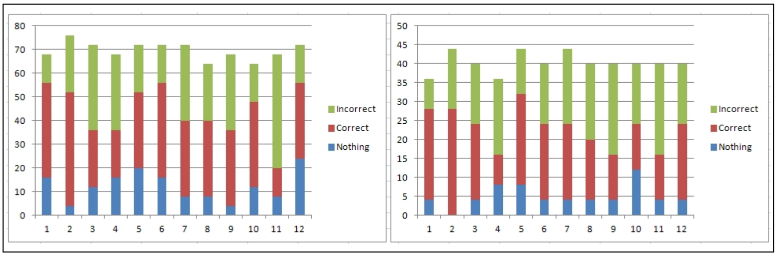


Figure 5: Items

We can analyse the students' results of each option of any item in the item bank in order to determine the strong and weak points in the students knowledge. Figure 6 presents results for the options (a, b, c, . . ., n) of the first item in the item bank.

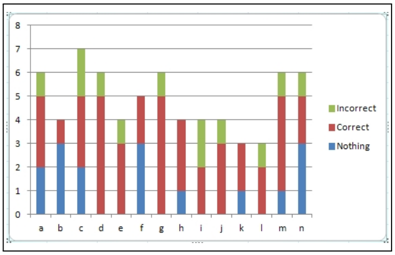


Figure 6: Options

Conclusion

We presented software tools for generating and checking multiple-choice tests of special type are publicly available and open source (see [3] and [4]). They are written in C++ using Qt – cross-platform application and UI development framework.
The tools are used in undergraduate programming courses for several years. They save a lot of time and effort of the teacher for the preparation, verification and evaluation of test results. Using the textbook and course materials during the test and impossibility of cheating combine evaluation of knowledge with elements of learning process.

References

[1] N. Kirov. A System for Assessing the Knowledge and Skills of Students in Computer Programming. In: Proceedings of the 9th Annual International Conference on Computer Science and Education in Computer Science, 29-30 June 2013 in Fulda, and 1-2 July 2013 in Wurzburg, Germany (ISSN 1313 8624), 109-112.

[2] W. L. Kuechler, M. G. Simkin. How Well Do Multiple Choice Tests Evaluate Student Understanding in Computer Programming Classes? Journal of

Information Systems Education, Vol. 14(4), (2003), 389-399.

[3] Test generator. `https://github.com/nkirov/tests_generator`

[4] Test checker. `https://github.com/nkirov/tests_checker`

Authors' Information

- ***Nikolay KIROV***
- ***PhD, Assoc. Prof.***
- ***New Bulgarian University***
- ***1618 Sofia, Montevideo str. 21, nkirov@nbu.bg***
- ***Programming, astroinformatics, digitization, image processing***