Sixth International Conference FMNS-2015
10-14.06.2015, Blagoevgrad, BULGARIA

Mini-conference "Inquiry-Based Approach in Higher Education in Mathematics and Informatics"

Test system and software for evaluation the students knowledge in programming

Nikolay Kirov

Computer Science Department, New Bulgarian University, Sofia, Bulgaria
`nkirov@nbu.bg`

Introduction

- Best practice, better practice, good practice, no so bad practice, . . .

- A test system for evaluation the students knowledge in programming.
- The test system as a part of the learning process of programming.
- The test system as a Inquiry-Based Approach in Higher Education in Informatics.

- For students: preparing for the test, test, verifying the results.
- For teachers: preparing the test, checking the test, analyzing the results.

Terminology

- The test consists of **items**.
- The **stem** is the introductory question or statement at the beginning of each item.
- The stem is followed by the **options**.
- The options are:
  - **answers** – the correct options, and
  - **distractors** – the incorrect options.
- The items are stored in an **item bank**.
- **Individual test** consists of fixed number of items with fixed number of options.

Special multiple choice test

- The individual tests are generated from the test bank randomly.
- The number of options in any individual test is fixed on 4 for all items.
- In any individual test the number of correct options (answers) of an item may be **any number** in the interval [0,4];
- The student have to identify each answer and each distractor. For any option he/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.
- Any correct response (yes or no) **adds** one point in the total score but any incorrect (oposite) response **subtracts** one point (a penalty point) from the total score.

Special multiple choice test

- We estimate an individual test of `M` items with `x` points total score as follows:
```
int p = ceil(100.0*x/(M*4)), e = 2;
     if (p >= 90) e = 6;
else if (p >= 76) e = 5;
else if (p >= 60) e = 4;
else if (p >= 50) e = 3;
```

- To calculate the probability of **passing** the test using random method, we choose a test with `M = 10` items (maximum 40 points total score).
- Probability is 0.11% in case the student has noted all the options **yes** or **no**.
- Probability is 0.0034% in case the student marked options randomly with **yes**, **no** or **nothing**.

Special multiple choice test – procedure

Before the test time:

- 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 [w].
- A few days before the test, a general advice is organized on which the issues of the test are discussed.

During the completion of the test [p]:

- The students can use lectures, textbooks and any other printed materials.
- The students are allowed to use computer as a book, or also compiler, or even Internet.
- Anyone can ask a question about ambiguities in the test.

Special multiple choice test – procedure

After the verification of the tests [p]:

- The individual tests are returned to the students.
- Each student should carefully check his/her individual test in order to determine whether he/she agrees with the noted errors.
- If something is not clear she/he can discuss the case.
- It is normal to increase the total score of the student if his/her arguments about the case are reasonable.

Why this type of test system is suitable for students in programming?

Inquiry-Based Approach

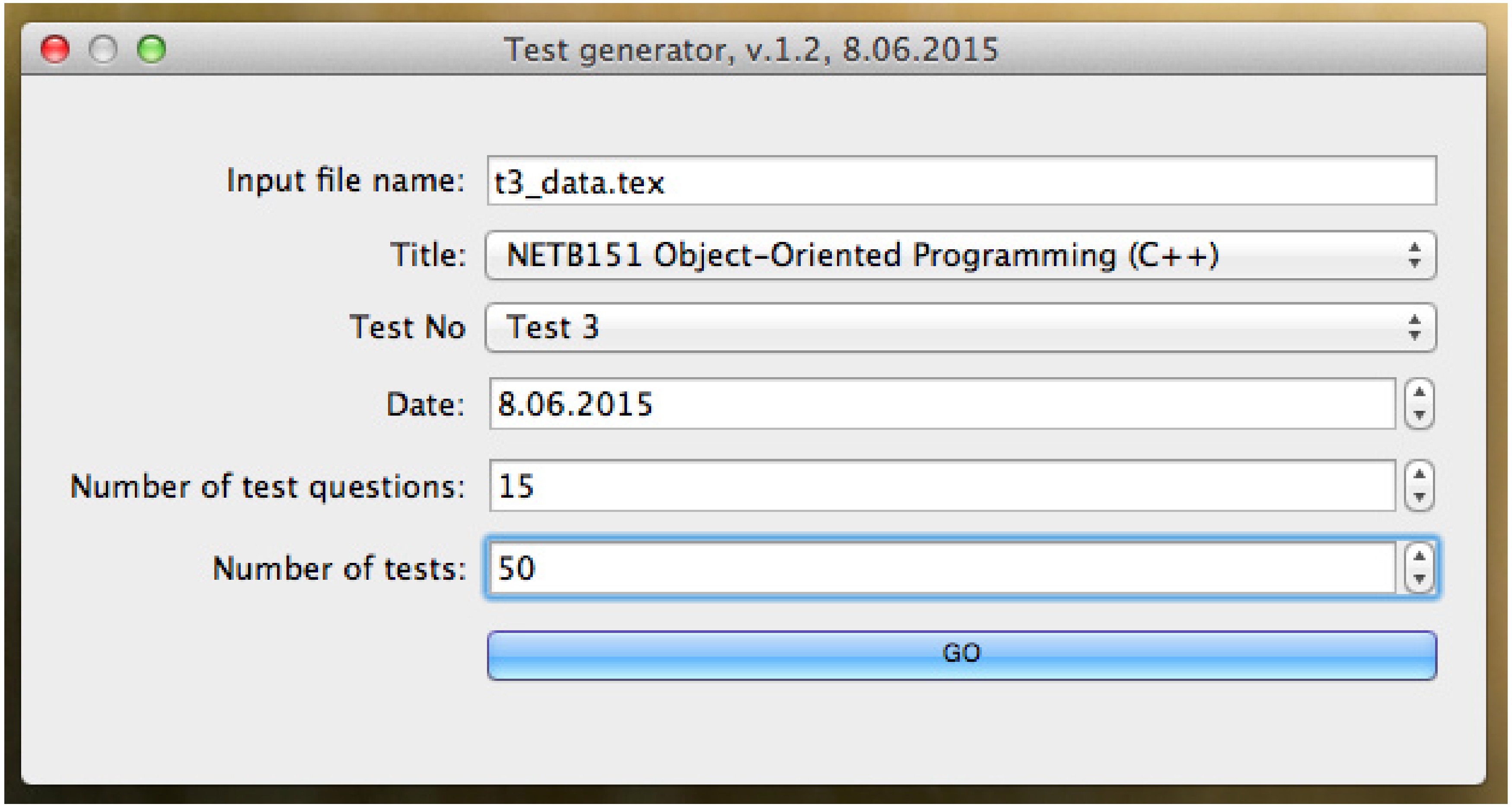
According to a definition by Linn, Davis and Bell, **inquiry** is the intentional process of diagnosing problems, critiquing experiments, distinguishing alternatives, planning investigations, researching conjectures, searching for information, constructing models, debating with peers, and forming coherent arguments.

inquiry		B	D	A
diagnosing problems	understanding the item	+	+	+
critiquing experiments	using compiler	+	+	-
distinguishing alternatives	comparing options	-	+	+
planning investigations	How to search?	+	+	-
researching conjectures	yes, no, nothing	+	+	-
searching for information		+	+	+
constructing models	programming	+	-	-
debating with peers		+	-	+
forming coherent arguments		-	-	+

Test generator – `github.com/nkirov/tests_generator`

- Preparation of the test begins with selecting items – stems and options and put them into **item bank** [f].
- At least 10 items each having at least 5-6 possible answers should be completed and stored as a text file in a particular format. This file is the input to `tests_generator`.
- `tests_generator` generates **individual tests** using random distribution of both items and their options [p].
- Each individual test consists of 10-20 items with 4 options (a, b, c, d).
- The output plain text file (`out.tex`) (in LaTeX format) contains all individual tests [f].
- The second output file (`tab.tex`) is a table for checking the tests [f].
- The third file (`data.tex`) is a copy of the input file with additional data for the generated individual tests [f].

Test generator – `github.com/nkirov/tests_generator`


Test generator, v.1.2, 8.06.2015

Input file name:	t3_data.tex
Title:	NETB151 Object-Oriented Programming (C++)
Test No	Test 3
Date:	8.06.2015
Number of test questions:	15
Number of tests:	50

Test checker – `github.com/nkirov/tests_checker`

Checking test can be carried out:

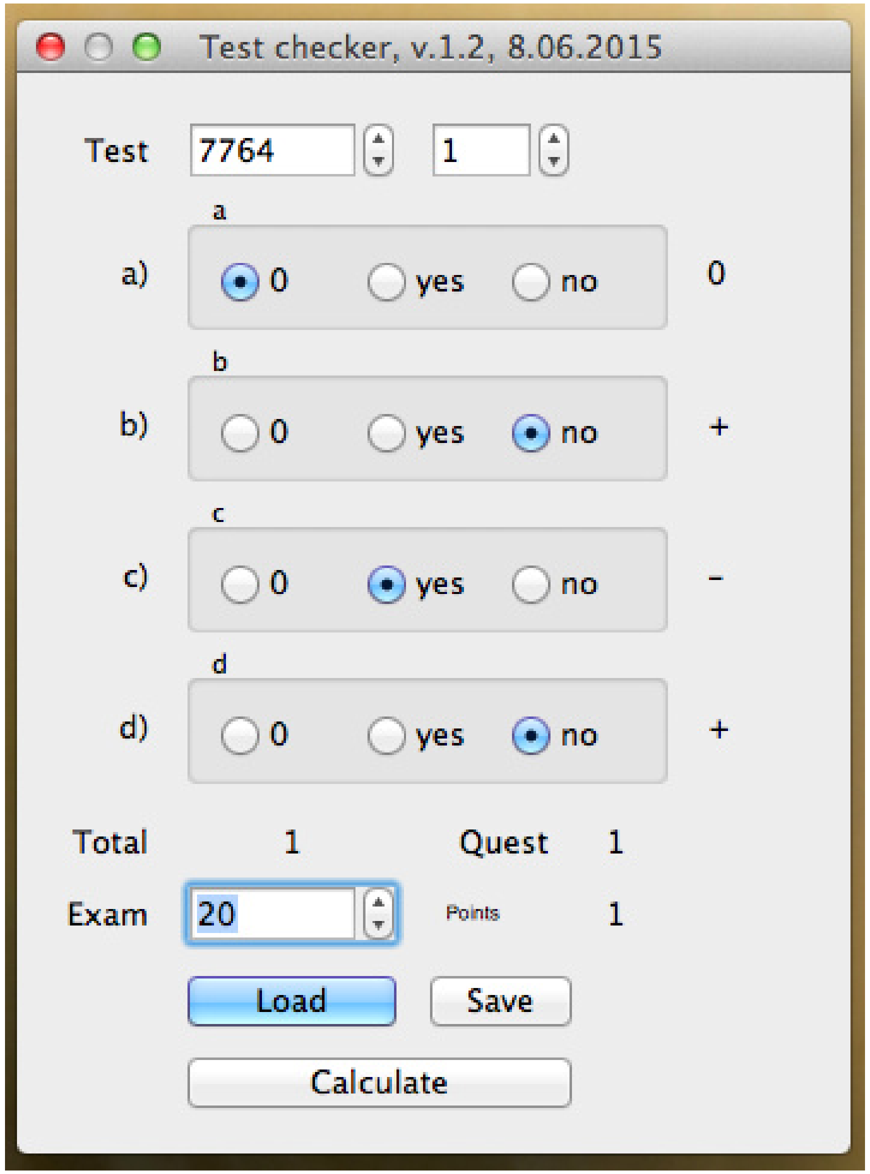
- manually – using the table (`tab.tex`) generated by `tests_generator` [p] or
- automatically – by `tests_checker`.

Uploading data on the students' answers can be done:

- manually – using the user interface of `tests_checker` or
- automatically – using a special template for students' answers and scanner [p].

The program creates a text file (`save.txt`), containing audited tests [f].

Test checker – `github.com/nkirov/tests_checker`


Test checker, v.1.2, 8.06.2015

Test	7764	1	
a)	[✓ 0] yes no		0
b)	0 yes [✓ no]		+
c)	0 [✓ yes] no		-
d)	0 yes [✓ no]		+
Total	1	Quest	1
Exam	20	Points	1

Test checker – `github.com/nkirov/tests_checker`

After entering the students' answers, `test_checker` gives the results – for each option of each item in the test bank calculates two sets of numbers (file `data_result.txt`) [f]. 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 individual tests of students, which pass the test (`e > 2`).

- a, b – the number of individual tests which contain the corresponding item and four of its options;
- a_1, b_1 – the number of tests without response;
- a_2, b_2 – the number of tests with correct response;
- a_3. b_3 – the number of tests with incorrect (opposite) response.

$a_1 + a_2 + a_3 = a$, $b_1 + b_2 + b_3 = b$, $b \leq a$ and $b_i \leq a_i$ for $i = 1, 2, 3$.

The output files of `test_checker` are `data_result.txt` and `data_result1.txt`, the format of the second file is suitable for input in spreadsheet.

Analysis of the test – example

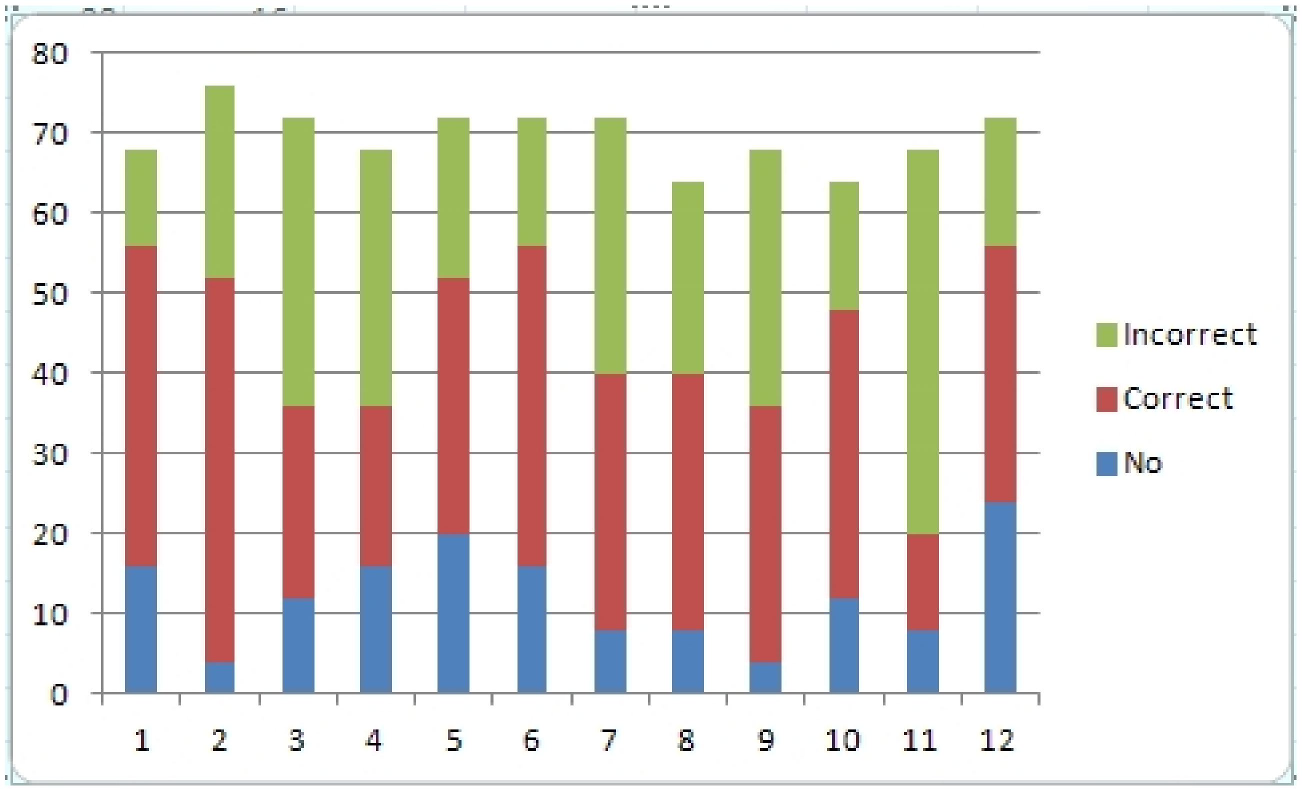
First test of Object-oriented programming (second semester), NBU, program "Network Technologies"

- Test bank consists of 12 item.
- The numbers of options are: (14,16,19,16,12,7,12,16,10,9,16,22).
- Any individual test consists of 11 items (maximum 44 points).
- We have 19 individual tests.

Analysis of the test – example

Items for 19 individual tests – X items, Y results.

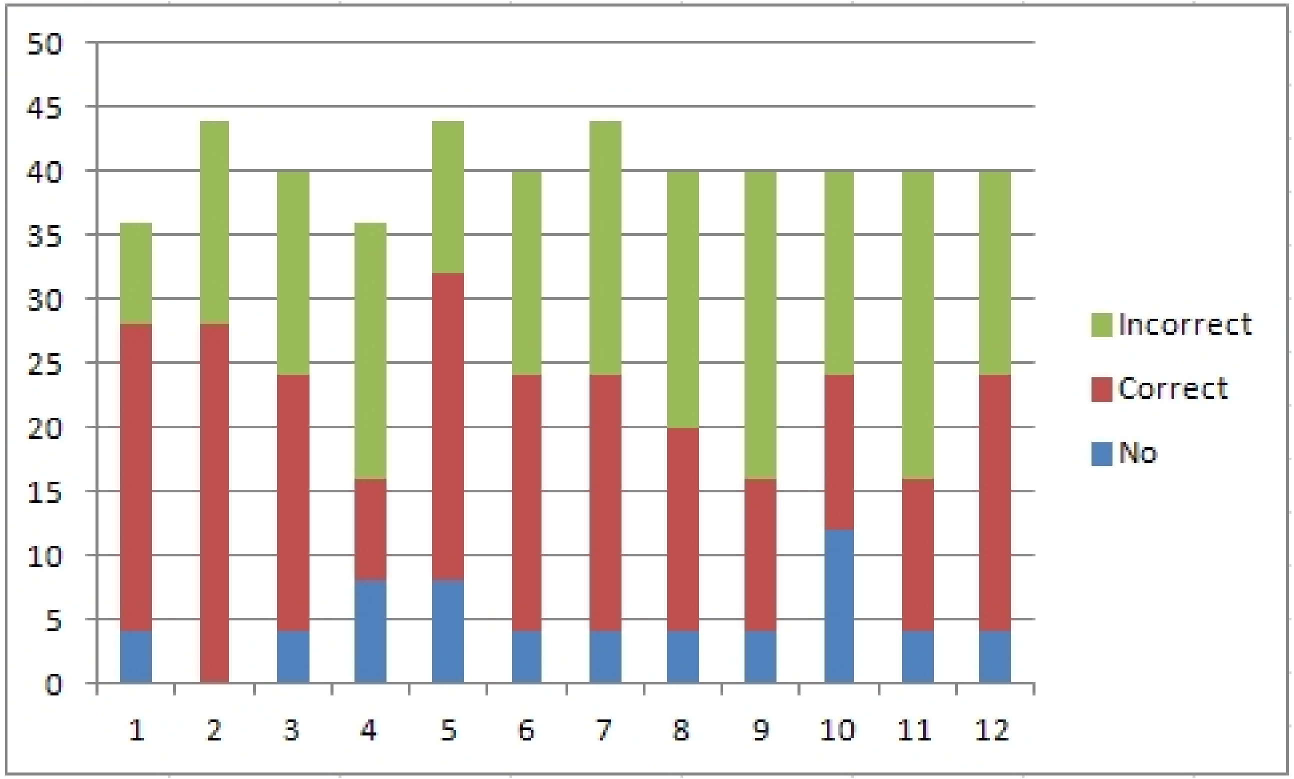
- Incorrect – the number of tests with the incorrect (opposite) response.
- Correct – the number of tests with the correct response.
- No – the number of tests without response.



Analysis of the test – example

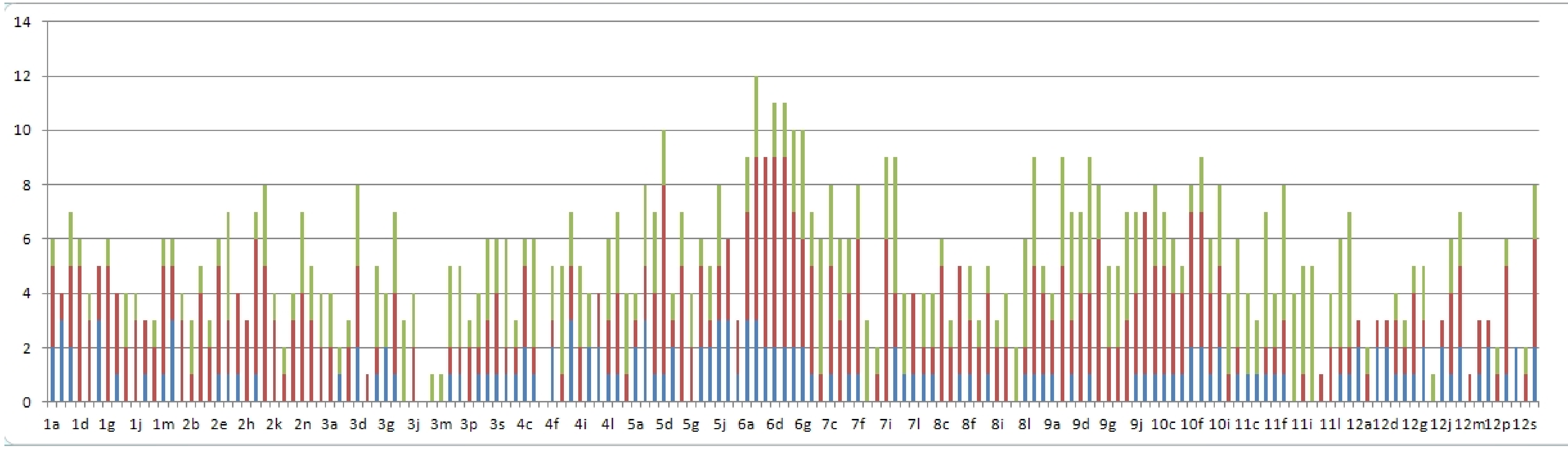
Items for 11 individual tests, which are collected at least a half (22) of maximum points (44).

- Incorrect – the number of tests with the incorect (opposite) response.
- Correct – the number of tests with the correct response.
- No – the number of tests without response.



Analysis of the test – example

Options for 19 tests: X – options, Y – results.



Conclusion

The software is written in C++ using Qt – cross-platform application and UI development framework (`qt.digia.com/`). It is publicly available and **open source**:

- `github.com/nkirov/tests_generator`
- `github.com/nkirov/tests_checker`

- The software tools save a lot of time and efforts of the teacher for the preparation and verification the test and evaluation the test results.
- The idea of such a test system arose in 1998, when I started teaching programming in Pascal for students from South-West University “Neofit Rilski”. Then I wrote the first version of `tests_generator` (in Pascal).

github.com/nkirov/tests_generator github.com/nkirov/tests_checker

Thank you for your attention.

nikolay.kirov.be/zip/nkk_edu_presentation_2015.pdf

nikolay.kirov.be nkirov@nbu.bg