Informatics in Education, 2009, Vol. 8, No. 2, 251–260

Examiner's Remarks on Informatics Matura Examination in Poland

Ewa KOLCZYK
Institute of Computer Science, University of Wroclaw
Joliot Curie Str. 15, 50–383 Wroclaw, Poland
e-mail: eko@ii.uni.wroc.pl

Received: July 2009

Abstract. The first specification for the informatics Matura examination in Poland was published in 2000, and since May 2005 the examination has been organized every year. This article includes some reflections and remarks about formulating examination tasks and pupils' difficulties in solving the tasks collected by the author during her work as an examiner. In the article, four examination tasks from 2008 are considered. These remarks could be useful especially for informatics teachers.

Keywords: informatics education, informatics teachers, informatics examination.

1. Introduction

The qualification system in Poland consists of several types of external examination. The role of the system is to ensure the comparability of learning outcomes. The external forms of assessment conducted at every educational stage are intended to help students, parents, teachers, schools, and education authorities to make clear which schools exemplify the appropriate level of teaching. The results of these external examinations are accepted as part of the requirements for schools in the next stage in admitting candidates. The results of the Matura examinations, taken at the end of secondary education, replaced entrance examinations to universities, and to other institutions of higher education. Every pupil who planning study at the university must pass three mandatory Matura examinations. Beside this, they can choose one or more additional examinations. The mandatory examinations can be taken at basic or advanced level, the additional examination – at advanced level. Pupils can choose both the subject and the level of examination. Their choice depends on what they want to study and the entrance requirements set by universities. The results of examinations are given in percentages and 30% is necessary to pass a mandatory examination.

The informatics Matura examination could not be chosen as a mandatory examination till 2009, so students took it as an additional examination at advanced level. Some of them wanted to check their skills and many of them supposed that it would be one of the easier examinations. In practice, the informatics Matura examination is a very difficult examination. The average results of this examination are the lowest among other Matura

Table 1
Number of pupils and their average result of informatics Matura examination

Year	Number of pupils	Average result (%)
2005	4498	28
2006	3222	23.9
2007	2079	37
2008	1547	36
2009	322	40.5
	1782	35.2

examinations. Institutions of higher education are very careful in setting their entrance requirements based on the informatics Matura examination results. Only a few institutions allow this qualification as an additional element of their entrance requirements. Computer Science departments at universities mainly require candidates to pass a mathematics Matura examination as a continuation of the traditional entrance examination in mathematics.

The difficulty of the informatics Matura examination and the limited interest from the higher education institutions' side lead to the number of pupils taking this examination to decrease.

This year, the situation has changed. The informatics Matura examination can be taken as a mandatory examination at two levels (basic or advanced). The new regulation has influenced pupils' choices. In Table 1 the first row for year 2009 concerns the basic level and the second row the advanced level.

2. Scheme of Assessment in Informatics Matura Examination

The informatics Matura examination consists of two units. The first one is a theoretical unit, with no use of a computer. It takes 90 minutes, and the student has to deal usually with three tasks. Two of them have short answer questions or extended response questions form, and are connected with formulating and analyzing algorithms. The third task consists of some true-false questions, which concern different topics.

Unit 2 takes 150 minutes and students work on practical tasks, using computers. The practical tasks are usually of the following types:

- a task, which requires programming skills (using programming language to implement simple algorithms, to read data from the file and to write the results back to the file),
- a simulation task, which could be solved using spreadsheet or programming language,
- a database task, which requires manipulating data using database system (or at least a spreadsheet).

The student has limited time to analyze the problem, to choose an appropriate tool, and to solve the task. In several tasks, the examiners receive and mark the results only. The correct results could be obtained in a set time, if you know what to do, and you are clever in using the appropriate software. The limited time is, equally, both an advantage and a disadvantage. It is possible to find which students are clever in choosing the best tool and in thinking quickly and precisely.

3. General Remarks for Teachers

In both units the tasks require careful reading, analyzing the conditions and choosing the proper method of solving the task.

In unit 1, the character of the task, consisted of true-false questions, make it easy to guess the correct answer, but every year just this task shows pupils' difficulties in careful reading and lack of knowledge of some notions. The teachers are responsible for checking if all notions mentioned in the subject criteria are known to pupils. Another reason of difficulty with true-false questions is that they are rarely present in informatics lessons. Maybe teachers should use them more often to evaluate pupils' understanding of some definitions and notions.

In unit 2 the proficiency in using a proper tool (spreadsheet, programming language or database system) is needed. Some pupils failed because they are not so clever in one or more mentioned domains, or they need more time to do the tasks. On the other hand many pupils leave the examination room before the end of the set time. It is possible that they admit defeat too quickly. Teachers should motivate pupils to undertake more trials of practice in solving a task and convince them, that it is enough to solve only a part of the task to obtain some points. In the classroom the pupils should practise independent solving tasks instead of doing tasks according to precise teacher's instruction, only.

In practical tasks in unit 2 the pupils often do not bother to put the correct answer in the properly named file or do not attach the files with their solution. Those cases make examiners unable to mark such solutions. It is very important that pupils should get used to prepare their solutions exactly according to the requirements given in the task.

4. Examples of Examination Tasks

The following examples of tasks are from the informatics Matura examination conducted in 2008. The chosen tasks are representative and pupils' difficulties in solving them are the source of important information for teachers. There are two tasks from unit 1 (the true-false questions are omitted) and two tasks from unit 2 (the simulation task is omitted).

The easiest part of the task from Table 2 was section a). This part was strongly connected with math curriculum because definition of integer sequence and its properties are known from math lessons. The section a) serves checking if pupils understand the given definition of integer sequence. Observing values of the first elements of the sequence should help pupils in discovering the regularity of the sequence, so they will be able to

Table 2

Powers – task from unit 1 (Matura examination, May 2008)

In the table below there are powers of two:

k	0	1	2	3	4	5	6	7	8	9	10
2^k	1	2	4	8	16	32	64	128	256	512	1024

The sequence $a = (a_0, a_1, a_2, ...)$ is defined as follows:

$a_k =$ remainder of division 2^k by 10 for $k = 0, 1, 2, \ldots$

a) Using given definition calculate first 16 elements of the sequence a. Put the results in the table below:

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a_k																

b) Using list of steps, flow chart or programming language write an algorithm, which computes the remainder of division 2^k by 10 for non-negative integer number k. E.g., for $k = 15$ the result of your algorithm's performance should be 8.

Specification:

Input: Integer number $k \geqslant 0$.

Output: Remainder of division 2^k by 10.

c) Using list of steps, flow chart or programming language write an algorithm, which computes a^n, where a is integer number, and n is power of 2 ($n = 2^k$ for integer number $k \geqslant 0$).

Suggestion: Notice that $a^n = a^{\frac{n}{2}} \cdot a^{\frac{n}{2}}$, for $n > 1$.

Specification:

Input: Integer numbers a and n, where $n = 2^k$ for integer number $k \geqslant 0$.

Output: The number $p = a^n$.

Important: Marking your solution depends on the correctness and the complexity of your algorithms. It means that your algorithms should perform possibly the smallest number of arithmetic operations (addition, subtraction, multiplication and division).

write the optimal algorithm in the section b). The optimal algorithm could be written as follows:

step 1: *if k = 0 then the result is 1*
step 2: *if* $k \neq 0$ *then:*
step 2.1: *calculate remainder of division k by 4*
step 2.2: *if remainder = 0 then the result is 6*
step 2.3: *if remainder = 1 then the result is 2*
step 2.4: *if remainder = 2 then the result is 4*
step 2.5: *if remainder = 3 then the result is 8.*

Many pupils use the definition of the sequence and write the algorithm according to the formula, so the steps of algorithm are:

step 1: *calculate the power of 2 and exponent equal k*
step 2: *calculate* 2^k *mod* 10
step 3: *put calculated value as the result.*

They do not reflect that exponentiation is not an elementary operation for computers and computing value of power of two requires some multiplication operations to do. The number of the operations is increasing when you take bigger values of k. Some of the

pupils do not know the MOD operator or use the DIV operator instead of. These operators are often used in programming languages and spreadsheets, so pupils are expected to know them. They should also remember the idea of integer division from math lessons.

Finding the regularity in the sequence causes that you do not need to calculate 2^k for every element of sequence, but only remainder of division k by 4. In this case the algorithm performs the same number of operations for every value of k. Of course if you want to complete the task correctly from mathematician's point of view, you should prove that the sequence defined in the task and the sequence described with the algorithm are the same sequences. The proof is not complicated, but it is not required in the examination sheet.

The task is a nice example of the way, how we can correlate mathematics and informatics. The definition of the notion of integer sequence is a part of math curriculum. Informatics could supply the tools (programming system or spreadsheet) for calculating elements of the sequence. Observing values or making the graph could help pupils to notice some properties of the sequence. They can formulate some conclusions and then try to prove them.

Finding algorithms and making them faster, e.g., by reducing number of performed operations is a part of informatics curriculum. Pupils could compute some results according to mathematical formulas, but they should reflect, if it is possible to do this computation faster. This reflection should be excited in informatics lessons, in the task in Table 2 the statement in the bottom reminds about such approach.

The section c) refers to an algorithm, which should be known for pupils. Exponentiation could be described as repeated multiplication, so pupils are expected to write a simply algorithm using iteration, e.g.:

step 1: *p := a*
step 2: *while n > 1 do:*
    step 2.1: *p := p * a*
    step 2.2: *n := n - 1*
step 3: *the result is p.*

In this algorithm the multiplication operation is performed $n - 1$ times, so depends on the value of the exponent n. The authors of the task try to encourage pupils to think of the better algorithm in which multiplication operation could be performed less then $n-1$ times. If you take into consideration the suggestion (see the Table 1) and that n could have only particular values ($n = 2^k$ for integer number $k \geqslant 0$), the algorithm could be written as follows:

step 1: *p := a*
step 2: *while n > 1 do*
    step 2.1: *p := p * p*
    step 2.2: *n := n div 2*
step 3: *the result is p.*

In the section c) the pupils are expected to know and to apply classic algorithms for computing powers. Many pupils do not know even the first algorithm. Some of them do not understand what does it mean to write algorithm in given notation. It is important

for teachers that they should care about writing and analyzing simple algorithms during informatics lessons.

The task from Table 3 is mainly about analyzing written algorithms. It deals also with important idea of recursion. Majority of pupils manage to do correctly the section a), but it is still actual for teachers to care about improving pupils' ability of analyzing algorithms. The lessons about algorithms should be full of examples of such analysis done with a sheet of paper and pencil, with educational software which enables simulation of algorithm's performance.

The section b) and c) are more difficult mainly because they require math skills. It is another example of correlation between mathematics and informatics. In maths lessons we use declarative statements as definitions and we conduct reasoning about some properties of defined notions. In informatics lessons definitions often are formulated in procedural way. The transition from declarative to procedural form and back could be fruitful for better understanding of the notion.

In the section d) pupils are expected to modify the algorithm from section a) according to the given definition of 3-regular strings. Modification is one of the methods of investigation of the algorithm's performance. The teachers could invoke the need of an algorithm's modification by asking questions like "What happen , if . . .?", "What should be change, so . . .?"

In the task from Table 4 in section a) pupils should write a program, which involves reading a word from the file, changing the order of characters in the word and writing the result in the file. The algorithm of putting characters of the word in reverse order is a simply and often used algorithm. Some programming languages deliver the function responsible for this operation. Pupils should demonstrate the ability to implement simple iteration and communication with disk files in the chosen language. Many pupils used a spreadsheet to find the longest (shortest) password and it was the easiest way to obtain correct results. Some of them tried to create words with characters in reverse order also using a spreadsheet, but in this case the solution was much more time consuming.

The section b) was much more difficult for pupils because it requires implementing the algorithm which is a combination of the algorithm from previous section and the classic algorithm of checking, if the word is a palindrome. Generally pupils failed with this task because of the lack of proficiency in implementing simple algorithms in any programming language. In informatics lessons teachers should organize writing programs in limited time or encourage pupils to check the time needed to write program during their individual work at home. Also during informatics lessons it is important to show different ways of solving tasks. Using a spreadsheet is very useful in many tasks when you need to sort elements of the file, to find maximum or minimum, or to make a graph. So teachers could show examples of solutions done with the programming language only, with the spreadsheet only and using both of these tools.

The task from Table 5 is a typical task dealing with a simple relational database. The structure of given files suggests what tables should be defined and how the tables should be joined. Finding answers for four of five questions requires to use data from more than one table. So the ability of defining queries, which properly link data from different

Table 3

Words – task from unit 1 (Matura examination, May 2008)

Let $A = \{a, b\}$ means the alphabet consisted of two letters. The string over the alphabet A is defined as a finite sequence of characters from the alphabet, which length is bigger than zero, e.g.:

$$a,\ ab,\ aba,\ baba,\ aaaa$$

Let $|w|$ means the length of the string w, so $|aba| = 3$.

If w_1 and w_2 are the strings, then w_1w_2 will mean the concatenation of strings w_1 and w_2. E.g., for $w_1 = ab$ and $w_2 = aa,\ w_1w_2 = abaa$.

2-regular strings are defined as follows:

- every string consisted of one letter is 2-regular,
- if the string w is 2-regular, then the string ww is also 2-regular.

Other strings are not 2-regular.

The recursive function **2REG** (w) checks if given string w over the alphabet A is 2-regular.

Specification:

Input: string w, which consists of letters from the alphabet A.

Output: answer *YES*, if the string w is 2-regular;

answer *NO*, if the string w is not 2-regular

2REG(w)

step 1: if $|w| = 1$, then the result is *YES*

step 2: if $|w| > 1$ and $|w|$ is odd number, then the result is *NO*

step 3: if $|w| > 1$ and $|w|$ is even number, then:

step 3.1: divide the string w into two strings w_1 and w_2 of the same length and $w = w_1w_2$

step 3.2: if $w_1 \neq w_2$, then the result is *NO*

step 3.3: the result is the value of **2REG**(w_1)

a) Write parameters of all recursive calls of the function **2REG** and calculate the result for the following strings:

i. $aabbaabb$

ii. $aaaaaaaa$

iiii. $bbbbbbbbbbbbbbbbbbbb$

e.g.: for the string $w = abab$, parameters of all recursive calls of the function **2REG** and the result are:

$$abab \rightarrow ab \rightarrow NO$$

b) What is the length of 2-regular strings? Explain your answer.

c) How many 2-regular strings over the alphabet A with the length n $(n \geqslant 1)$ is? Explain your answer.

d) 3-regular strings are defined as follows:

- every string consisted of one letter is 3-regular,
- if the string w is 3-regular, then every string wxw, wwx, where x is the string over the alphabet A and $|w| = |x|$, is 3-regular string.

Other strings are not 3-regular. Examples of 3-regular strings are: $a,\ aba,\ abaabaaaa$. The string $aaaabaaba$ is not 3-regular.

Using list of steps, flow chart or programming language write an algorithm, which checks if given string w over the alphabet A is 3-regular.

Specification:

Input: string w, which consists of letters from the alphabet A.

Output: answer *YES*, if the string w is 3-regular;

answer *NO*, if the string w is not 3-regular.

Table 4

Passwords – task from unit 2 (Matura examination, May 2008)

The file `words.txt` contains 1000 words which length is not more then 30 characters. Every word is put in a new line and consists of uppercase letters.

a) The passwords are produced from words in the file `words.txt` by putting the characters of the word in the reverse order.

Example

Word	Password
KAJAK	KAJAK
EGZAMIN	NIMAZGE
MATURA	ARUTAM
KOMINIARZ	ZRAINIMOK

Create passwords from words in the file `words.txt` and put them in the file `passwords_a.txt` (every password in a new line). Find the longest and the shortest password, their lengths and put the results in the file `words_a.txt`.

b) Palindrome is a word that is the same whether you read it forwards from the beginning or backwards from the end. Create password from word w according to the algorithm described below:
- let $w = w_1 w_2$ means that the word w is the result of concatenation of words w_1 and w_2,
- find the longest word w_1, where w_1 is a palindrome in the beginning of the word w and $w = w_1 w_2$,
- the password is the result of concatenation of the words: w_2 written in reverse order and w.

Attention: If w is a palindrome then $w = w_1$, and the word w_2 is empty.

Example

Word	The longest palindrome in the beginning	Password
KAJAK	KAJAK	KAJAK
KAJAKARSTWO	KAJAK	OWTSRAKAJAKARSTWO
MAMA	MAM	AMAMA
KAKTUS	KAK	SUTKAKTUS
WANNA	W	ANNAWANNA
EGZAMIN	E	NIMAZGEGZAMIN

Create passwords from words in the file `words.txt` and put them in the file `passwords_b.txt` (every password in a new line). Find answers for the following questions and put them in the file `words_b.txt`.

1. Find all passwords which length is equal 12.
2. Find the longest and the shortest password.
3. Find the sum of the lengths of all passwords.

tables was the key skill for pupils. Many of them failed because they did not join the tables properly or they did not group records according to the conditions described in the task. Some of them failed because they used spreadsheet instead of database system (like MS Access). In spreadsheet you do not have mechanism of joining different tables. So it is important to show how to deal with data when they are not collected in one table.

Table 5

Car accidents – task from unit 2 (Matura examination, May 2008)

The insurance company collects following data about cars: **registration number**, **brand name**, **year of production**, **ID number of the owner** and following data about owners: **first name**, **last name**, **ID number**, **residency type**. Besides, the insurance company collects data about car accidents, which are caused by owners of the cars and about amounts of money the company had paid out as the covers for accidents.

In the file `cars.txt` there are data about cars: registration number, brand name, year of production, ID number of the owner.

Example

BAU1876 Skoda 1998 59042500616
BAU3353 Renault 1999 54010520609

In the file `persons.txt` there are data about owners: first name, last name, ID number, residency type. The residency type means:
A – city, B – town, C – small town, D – village.

Example

46073182890 Kornel Henrykowski A
46080423256 Jan Bugajski B

In the file `accidents.txt` there are following data: accident's number, date of accident, car registration number, amount of money, which the company had paid out.

Example

1	1996-01-03	BL24933	10453.00
2	1997-10-14	GCH9779	673.00
3	2002-03-24	NWE4941	8276.00

Find answers for following questions and put them into the file `answers.txt`. Put corresponding letter before every answer.

a) How many owners of the cars have one or more accidents. The owner, who take part in more than one accident should be counted once.
b) Find the car registration number, first name and last name of the owner, who got the biggest amount of money from the insurance company and how much it was.
c) Find sums of money, which the insurance company had paid out in the years 2006 and 2007.
d) Find a car brand name, which is reported in the biggest number of accidents and what was this number. If any car is reported more than in one accident, count every accident the car was involved.
e) Find numbers of accidents, in which the owners from different type of residency take part (separately for cities, towns, small towns and villages).

The weak results of pupils' work on this task are caused by the low consciousness of manipulating data collected in a database and the low proficiency in using any database system. So teachers should spend more time with students on doing exercises with different databases, when students learn how to import data from text file to the database system, how to define and join tables, how to prepare queries and finally how to export the results back to the text files. The queries should be simple, connected with choosing some records from one or more tables and more complicated, where the records should be grouped and the aggregate functions used.

5. Conclusion

The practical unit 2 was much more difficult for pupils then the theoretical unit 1. The reason is that in unit 2 besides reading and analyzing the questions carefully, students have to choose the tool and use it with a high level of proficiency. In the unit 1 the main difficulty was in insufficient preparation to specify problems and to formulate algorithms. Many candidates failed even the task required analyzing the written algorithm to find its result, or to identify the purpose of it. The teachers are advised to take care about doing more tasks in which the students have to analyze real problems and get more programming experience.

In the future, the value of this examination may change, but for today it could be considered as a useful activity in the secondary curriculum, and as a model for informatics education. The informatics Matura examination is not, and would not be, the popular and mass examination. It is addressed to the clever pupils, who are interested in informatics understood as broadly as is possible. Preparing for this examination should give these pupils a feeling about what is involved in studying informatics.

References

Informatics Matura Examination, Assessment Materials, Unit 1, May (2008).
http://www.cke.edu.pl/images/stories/Arkusze08matura/inform_cz_1.pdf
Informatics Matura Examination, Assessment Materials, Unit 2, May (2008).
http://www.cke.edu.pl/images/stories/Arkusze08matura/inform_cz_2.pdf
Informatics Matura Examination, Examiners' Reports, May (2008).
http://www.cke.edu.pl/images/stories/08_wyn/spr_mat_przyr_a.pdf

E. Kolczyk is working in the Institute of Computer Science, University of Wroclaw. She received her PhD degree in the Institute for Educational Research in Warsaw (2002). Her research interests are didactics of informatics, teaching informatics at primary and secondary level. She is co-author of educational materials for informatics in schools.

Tiriant Lenkijos informatikos brandos egzaminą

Ewa KOLCZYK

Pirmoji informatikos brandos egzamino analizė Lenkijoje buvo atlikta 2000 metais, o nuo 2005 metų gegužės tokie tyrimai buvo organizuojami kasmet. Šiame straipsnyje autorė pateikia svarstymus ir pastabos apie egzamino užduočių formulavimą ir mokiniams iškilusius sunkumus sprendžiant šias užduotis. Nagrinėjamos ir lyginamos keturios 2008 metų brandos egzamino užduotys. Pateikiama informatikos mokytojams naudingų pastabų.