XXVIII НАЦИОНАЛНА ОЛИМПИАДА ПО ИНФОРМАТИКА
5-6.05.2012 г.

Група А (12 клас)

Task A1. TETRIS

Despite its huge popularity, very few people know that the game Tetris is created by Russians. The goal in the game is to put falling figures in a two dimensional board (called "well").

Elly decided to show her friends how good she is in the game by reaching unimaginable records. Of course, playing the whole day would be rather untypical for her, so she wants to write an artificial intelligence that does that for her. It would be rather awesome to beat everybody else while being on shopping. She wants you to write the artificial intelligence that plays the game for her.

Even if you've never played the game, that's no problem – this will not handicap you in any way (the A.I. can be much smarter than any human). If you've played the game before, please read the exact rules, since we are going to use a slightly easier version of the game for convenience.

Scoring

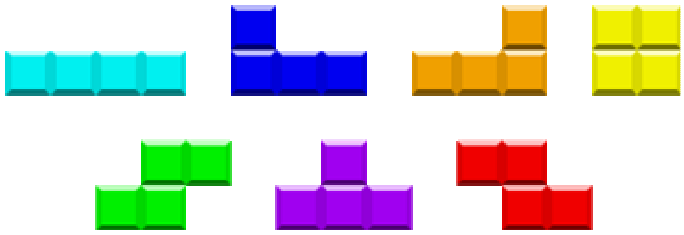
In contrast to the original game, the score here will be dependent on the number of figures that the player successfully puts in the well before the end of the game.

Board

The game is played on a board with 20 rows and 10 columns (thus, the depth of the well is twice its width).

Figures

As in the original game we will use 7 different figures, each of which consists of four blocks (where the name of the game comes from – the Greek prefix "tetra-", meaning "four"). The seven figures look as follows:



Which can also be represented in ASCII art:

Fig. 1	*Fig. 2*	*Fig. 3*	*Fig. 4*	*Fig. 5*	*Fig. 6*	*Fig. 7*
" # " " # " " # " " # "	" ## " " # " " # " " "	" ## " " # " " # " " "	" ## " " ## " " " " "	" # " " ## " " # " " "	" # " " ## " " # " " "	" # " " ## " " # " " "

The shapes of the figures when not rotated (rotation 0).

Rotations

Each of the figures can be rotated zero, one, two or three times 90 degrees clockwise. Note that a fourth rotation would lead to the original figure for any of them. For example, some of the pictures of the figures are obtained after one or more rotations of the original ones (shown in the ASCII art table).

Falling of a figure
The figures fall one after another. In each moment exactly one figure falls and the other "wait" in a queue somewhere outside the board. When the turn of a certain figure comes, it is put ABOVE the well (on top outside the board). The player chooses how it will be rotated and in which column will be its leftmost block. After that the figure starts falling vertically down until one or more of its blocks reaches the bottom of the well or another block from a previous figure already inside it. Note that the position of the figure must be chosen in such way, that the figure falls completely into the well (i.e. none of its blocks is outside the board to the left or to the right). If after the fall and the eventual following deletions some of the blocks of the figure are still outside the well (on top), the game ends.

Deletion of a row
After a figure falls down, one or more rows of the board might get "full", i.e. all ten columns of a row contain a block of a figure. If this is the case all the blocks in this row get deleted and disappear. This is done in ALL full rows before the game continues onward.

Components
After the deletion of a row there might be blocks hanging "in the air" (i.e. their support just disappeared). They are divided into connected components such that two blocks belong to the same component if there is a path from one to the other through another blocks only going up, down, left and right. See the examples further in the statement for clarification.

Falling of a component
After the deletion of one or more rows, all components hanging "in the air" start falling simultaneously down following the same rules as the figures – until one of their blocks reach the bottom of the well or a block of another, already static component.

End of the game
The game ends when some of the figures is put in such way, that after its fall and the following deletions one or more of its blocks is left outside the board. The game also ends if the player makes invalid move or when all figures are put successfully.

Examples
For clarification of the rules, take a look at the following example (only the bottommost ten rows of the well are shown).

Step 1	*Step 2*	*Step 3*	*Step 4*
" 2 "	" "	" "	" "
" 2 "	" "	" "	" "
" 22 "	" "	" "	" "
"11 3 "	"11 3 "	"11 5 "	" "
"11 333"	"11 333"	"11 555"	"11 5 "
"1 33 "	"1 2 33 "	"1 4 55 "	"11 555"
"11 33 "	"11 2 33 "	"11 4 55 "	"1 55 "
"111 33333"	"1112233333"	" "	"11 55 "
"111 333333"	"111 333333"	"222 333333"	"2224333333"
"111 3 3333"	"111 3 3333"	"222 3 3333"	"22243 3333"

The blocks of the different components are marked with different digits in the odd steps. In the even steps the digits are preserved (although the components are different) in order to show which block went where.

Step 5	*Step 6*	*Step 7*	*Step 8*
" "	" "	" 33 "	" "
" "	" "	" 33 "	" "
" "	" "	" "	" "
" "	" "	" "	" "
"11 4 "	" "	" "	" "
"11 444"	"11 4 "	"11 2 "	"11 2 "
"1 44 "	"11 444"	"11 222"	"11 222"
"11 44 "	"1 44 "	"1 22 "	"1 33 22 "
" "	"11 44 "	"11 22 "	"1133 22 "
"22222 3333"	"22222 3333"	"11111 2222"	"11111 2222"

Note that after the fall of the components in step 3, in step 4 there is a new full row, which gets deleted in step 5. This continues until all components become static without creating a full row. After this the game continues with the next figure, as it is shown in step 7.

In contrast with the original game, here the player knows in advance all figures that will fall (what is their type and in what order they will fall).

Input
On the first row of the standard input will be given a single integer *N* – the number of figures that will fall during the game. On the second line will be given *N* integers between 1 and 7, inclusive – the figures that will fall in the order of falling.

Output
On the standard output print *N* pairs of integers. The first number of each pair must be between 0 and 3, inclusive, giving the rotation of the current figure. The second number must be between 0 and 9, inclusive, giving the column where the leftmost block of the rotated figure resides before falling. Rotation 0 is considered the orientation of the figures as given in the ASCII art table.

Constraints
There will be 10 tests with 20, 50, 200, 500, 1000, 2000, 3000, 5000, 10000, and 10000 (again) figures, respectively. The figures will be chosen randomly (i.e. there will be no "specific" tests) – each figure has the same chance to be chosen at each move.

Scoring
If your program has put X out of N figures before the end of the game, you will receive round(X * 10.0 / N) points. Thus, if for example each test is worth 10 points and you have put 72 out of 100 figures, you will get 7 points for this test. As another example if your solution put 963 out of 1000 figures, you will get all 10 points.

Input
20 2 7 6 4 7 3 6 6 4 6 3 4 2 1 1 1 5 7 4 2

Output
2 1 0 0 0 0 3 0 2 4 1 7 0 6 1 7 0 6 3 7 2 3 0 2 0 5 1 8 2 4 2 4 2 4 2 4 2 4 2 4

*The first 12 figures form the board of the example. The 14-th figure is put in such way, that it gets outside the board (to the right), which is invalid move. Thus, the player made 13 moves before the end of the game, gaining round(13 * 10.0 / 20) = round(6.5) = 7 points.*

Clarification
First 11 figures:

```
"   B       "
"   B       "
"   BB      "
"44       A "
"44      AAA"
"3      99  "
"33     99  "
"321   57888"
"221  557786"
"211  5 7666"
```

Visualizer
For convenience and further clarification of the process of falling you are given the sources of a visualizer, which you can also use to test your output. In the directory of the executable must be given two text files: Tetris.in, which contains the input (the number of figures and the figures themselves), and Tetris.out, which contains the output of your program (N pairs of integers, giving the rotation and the column for each figure). As arguments to the visualizer you can provide the number of milliseconds between two steps of the game. If not provided, the default value of 100 will be used (i.e. the time between two consecutive moves is 0.1 seconds). If the argument is 0 the visualizer prints only the score.
You are allowed to use parts of the visualizer in your own solution!