Tasks of the third stage of the Logia16 contest

- the subject IT competition

for junior high school students of the Mazowieckie Voivodeship

March 9, 2016

Task 1 (trip).

Cities in Logolandia are numbered and connected by a specific network of roads. Each road connects two different cities. From any city you can reach any chosen one, beating one or more roads without going back - only in one way. One of the cities is distinguished and is called the capital. The capital is connected with the rest of the country at most two roads, and every other city - three at most.

Małgosia, Karol and Paweł visit all the cities of Logolandia during a common trip, without unnecessary passages. They set off from the capital and end the trip there. For example, let's assume that Logolandia looks like in the picture example, the capital city is number 4 and the route of the trip runs through cities 4, 2, 1, 2, 3, 2, 4, 5, 6, 5 and 4. Each participant Trip saves the route according to your own rules, but each city is saved only once.

Małgosia is saving the city during his first stay there. In the example subsequent cities saved by Małgosia are 4, 2, 1, 3, 5 and 6.

Karol records the city only when he can no longer go to a new one city, or when entering the city for the second time. In the example, subsequent cities saved by Karol is 1, 2, 3, 4, 6 and 5.

Paweł does things differently - he records the city only when he is in it for the last time. The next example in the example the cities recorded by Paul are 1, 3, 2, 6, 5 and 4.

Write a two-parameter `description` function , which results in route descriptions made by Małgosia and Karol there is a description of the route made by Paweł. Route descriptions are lists containing the numbers of the cities visited. The result of the function is a list containing city numbers prepared by Paweł.

Examples in the Logo:	*Examples in Python:*
the result of the `description [4 2 1 3 5 6] [1 2 3 4 6 5]` is `[1 3 2 6 5 4]`	the result of the `description ([4 , 2 , 1 , 3 , 5 , 6] , [1 , 2 , 3 , 4 , 6 , 5])` is `[1 , 3 , 2 , 6 , 5 , 4]` ,
the result of the `description [1 2 4 5 3 6 7] [4 2 5 1 6 3 7]` is `[4 5 2 6 7 3 1]` ,	the result of the `description ([1 , 2 , 4 , 5 , 3 , 6 , 7] , [4 , 2 , 5 , 1 , 6 , 3 , 7])` is `[4 , 5 , 2 , 6 , 7 , 3 , 1]` ,
the result of the `description [1 2 4 3] [4 2 1 3]` is `[4 2 3 1]` .	the result of the `description ([1 , 2 , 4 , 3] , [4 , 2 , 1 , 3]) is [4 , 2 , 3 , 1]` .

Task 2 (products).

Ania plays with numbers from 1 to 1,000,000. Each writes as a product of two numbers, but such whose difference is the smallest possible.

Write a single parameter function `of` a `number` whose parameter is a list of numbers and the result of a corresponding list of par numbers (two-element lists) created by Ania. In each pair, write the number so that the first one is not greater than the second.

Examples in the Logo:	*Examples in Python:*
the result of the `number [13 44 42]` is `[[1 13] [4 11] [6 7]]` ,	the result of the `number ([13,44,42])` is `[[1,13], [4,11], [6,7]]` ,
the result of the `number [12 7 10 24]` is `[[3 4] [1 7] [2 5] [4 6]]` ,	the result of the `number ([12,7,10,24])` is `[[3,4], [1,7], [2,5], [4,6]]` ,
the result of the `number [4 25 46 33]` is `[[2 2] [5 5] [2 23] [3 11]]` .	the result of the `number ([4,25,46,33])` is `[[2,2], [5,5], [2,23], [3,11]]` .

Task 3 (page).

The Antek list contains information about the views of its website. For each impression, in the two-element list, two data are stored: the moment of entry and the time the user has spent on the site. Data is saved in full seconds.

Write a one-parameter `max` function , whose parameter is the *Antek list* , and the result is the maximum number users watching the Antek website in the same second. As a user watching a page in a given second, We count everyone who in this second enters the site, is on it for some time, be in this second stops watching it.

Examples in the Logo:	*Examples in Python:*
the result of the `max [[1 2] [2 2]]` is `2` ,	the result of the `max ([[1,2], [2,2]])` is `2` ,
the result of the `max [[1 10] [2 8] [3 6]]` is `3` ,	the result of the `max ([[1,10], [2,8], [3,6]])` is `3` ,
the result of the `max [[1 2] [2 3] [1 10] [2 2] [5 5]]` is `4` .	the result of the `max ([[1,2], [2,3], [1,10], [2,2], [5,5]])` is `4` .

Task 4 (robot).

The robot moves on a square board consisting of `n` $_2$ fields (`n` lines and `n` columns, 1 ≤ `n` ≤ 1 000). He can only be found once in each field. His initial direction is the "right" direction. In every the next move moves the robot to move without changing its direction. If it is impossible (because he would leave the board or enter the field already visited) - once it turns right 90 ° and tries to go on. However, if it can not after the turn move on (because he would leave the board or enter the field already visited), it stops.

Write a three- `parameter` lnp function whose value is the number of fields on the board that the robot will not visit. The first parameter `n` specifies the size of the board. The second and third parameters (numbers from 1 to `n`) specify the row, respectively and the robot's initial position column.

Examples in the Logo:	*Examples in Python:*	
the result of `lnp 4 1 2` is `4` ,	the result of `lnp (4,1,2) is 4`	- the robot will visit only the fields located at the edges of the board,
the result of `lnp 4 3 1` is `8` ,	the result of `lnp (4,3,1) is 8`	- the robot will visit only the fields in the last two rows of the board,
the result of `lnp 4 4 4` is `15` .	the result of `lnp (4,4,4) is 15`	- the robot will not make any move.