NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER ENGINEERING
COMPUTER AND COMPUTER TECHNOLOGY SECTOR

Spring 2021

PROGRAMMING LANGUAGES I.

Exercise 3

Closing date and time of electronic submission: 18/7/2021, 23:59:59

Sort by tail and stack, again (0.25 points)

The problem with sorting numbers using a queue and a stack is known from the previous series of exercises this year. The question of this exercise is to write its solution in Java. Your program should have the same behavior with the Python programs you delivered for the second set of exercises. For examples of second-line pronunciation, exit your program should be as follows (note that this means that your main class - the one with the method main - should be called " **QSsort** ", pay attention to uppercase and lowercase letters):

$ java QSsort qs1.txt
QQSQSSQQSS

$ java QSsort qs2.txt
QQQQSQSSSQSSQS

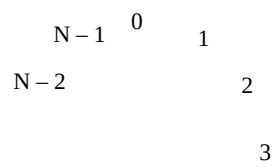
$ java QSsort qs3.txt
QQSQQSQSSS

$ java QSsort qs4.txt
QQQQSSSS

$ java QSsort qs5.txt
empty

All around (0.25 + 0.25 + 0.25 = 0.75 points)

Along a circular road are located N cities, numbered from 0 to N − 1, as in the adjacent figure. The vehicles can only move clockwise, ie from city 0 to city 1, from city 1 to city 2, and so on, and closing the circle from city N – 1 to city 0. On this road there are K vehicles, each of them which happens to be initially located in one of the cities. In every city can be located none, one or more vehicles.



Whenever a vehicle moves from the city where it is located in the next along the circle, we say that a "movement" takes place. We consider that vehicles cannot move at the same time, ie that the "movements"

are made in order, one after the other. A series of moves we say is "legal" if not the same vehicle moves twice in a row, without moving another vehicle in between.

Given the N, K and the initial positions of the vehicles, we want to find a legal one row consisting of the least possible movements, so that at the end of all vehicles be in the same city.

The exercise asks you to write three programs (one in ML, one in Prolog and one in Java) which read the input as shown below and find the minimum number M movements needed to find all vehicles in the same city as well the number of the city C they will end up with. If there are more legal series

Page 2

with the least number of moves, your program should find the lowest possible value of C. You can assume that the input will be such that the problem has a solution (note that there are cases for which there is no solution to the problem).

The input data will be read from a file in a format similar to that shown in examples below. The first line of the file will contain two integers separated by a space: the number of cities **N** ($1 \le N \le 10,000$) and the number of vehicles **K** ($2 \le K \le 10,000$). The second line will contain K integers numbers divided by two with a space, which will represent him in order number of the city where each vehicle is originally located.

Your programs, depending on the language of their implementation, should behave as in the following examples. For all implementation languages except Prolog, results are printed in standard output.

In SML / NJ

```
- round "r1.txt";
6 3
val it = (): unit
- round "r2.txt";
3 0
val it = (): unit
```

In MLton or in OCaml

```
./round r1.txt
6 3
./round r2.txt
3 0
```

In Java

```
$ java Round r1.txt
6 3
$ java Round r2.txt
3 0
```

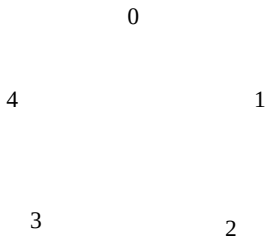
In Prolog

```
? - round ('r1.txt', M, C), write (M), write (''), writeln (C), fail.
6 3
false .
? - round ('r2.txt', M, C), write (M), write (''), writeln (C), fail.
3 0
false .
```

where the files with the input data are as follows (the command **cat** is a Unix command):

```
$ cat r1.txt
5 4
2 0 2 2
```

```
$ cat r2.txt
3 4
2 0 2 2
```



The initial position of the vehicles for the first example is shown in the adjacent figure. All vehicles can be found later of 6 moves in the city 3. To do this, the vehicle that is originally located in city 0 can move alternately with one

from vehicles originally located in the city 2.

Further instructions for the exercises

- You can work in groups of up to two people. You can if you want to form different group compared to the previous series of exercises - the groups in the system submission is anyway new to each series of exercises.
- You are not allowed to share your programs with fellow students outside your group or put them in a place where others can find them (eg on a web page, on talk sites,…). In case "strange" similarities are observed in programs, the grade of the students involved in *all the series of exercises* is done automatically zero regardless of which team ... was "inspired" by the other.
- You can use "helper" code (eg some code that it manages some data structure) that you found on the internet in your programs, with

Page 3

provided that your program contains in the comments the assumption about the origin of this code and a link to it.

- Programs in Python must be in a file and work in Python 3.7.3.

  (Note that Python 2 is a different dialect of Python!)
- Programs in Prolog must be in a file and work on one of them following systems SWI Prolog (8.0.2), GNU Prolog (1.3.0) or YAP (6.2.2).
- The program code in Java can be found in more than one file if you want, but it should be able to be compiled without problems with Java compiler with commands of the format: **javac QSsort.java** and **javac Round.java** . Do not set your own packages! Your Java submission can be either a single **.java** file or consist of a **.zip** file of a directory that contains **.java** your submission files (and only these - do not submit **.class** files). In every In this case, your submission must have a file with the names shown above in this paragraph.
- The submission of programs will be done electronically through moodle, as in previous exercise, and to be able to submit them, your team members (and both) should already have a moodle account. There will be a relevant announcement as soon as the submission system becomes active. Your programs should read it input as stated and must not have any other type of output other than requested because they will not be accepted by the submission system.