Big Homework 1
Data Structures and Algorithms Stacks, Queues & Lists

General mentions

- For this project you will work either in **teams of 2 people or alone**.
- The homework will be uploaded on the **Moodle platform (curs.pub.ro)**, for Hw1 assignment.
- If you encounter problems with the platform, contact the lab assistant responsible for that problem.
- The homework must be submitted by **25.04.2020, at 23:59**. No late submissions will be accepted.
- You will be asked details about your solutions during the following lab.
- The final submission will contain an archive named ***Student1FamilyName_Student1Name_ Student2FamilyName_Student2Name_HW1*** with:
    - the **source files of your project (.cpp and .h)**, grouped in separate folders for each exercise (and not contain the object files (.o) or executables (.exe) or codeblocks project files (.cbp))
    - a **README file** in which you will specify all the functional sections of the project, together with instructions for the user; additionally, if you have parts of the homework that don't work, you may offer solution ideas for a partial score on these sections.
- Warning: we will use plagiarism detection software on your submissions (Stanford's tool Moss). Copied homework will be marked with 0 points.
- Observation: The stacks, queues and lists used will be in headers (.h) and will be generic classes (template).

Ex1 - Stack (3p)

You are Ethan Hunt and IMF has sent you on a mission to protect the nuclear engineer specialist, Homer Simpson, who is in town for a very important presentation from a possible kidnap by the evil corporation called The Syndicate. You are gathering the information about your target but you soon discover that the IMF has no image of him, so you don't know what your target looks like. *What you do know is the fact that he is a reputable specialist and everyone will talk to him. What you also know is the fact that Homer is a really shy guy and he will not try to talk to anyone in return because he doesn't know anyone from the event.* Tonight, there is a formal dinner for all the participants and you are sent to gather information about the interaction between people in order to identify your target. After the event, you take your notes back to the base camp and you look over all the interactions that have happened between people under a matrix of N x N form, where 0 means that the person i has not talked to person j and 1 means that person i has talked to person j. Using a Stack, find your target and save him from the kidnapers.

P.S. It might be possible that he was so shy that he decided to skip the dinner. If this happens, then you need to create another event in order to gather again information

Input Example

```
MATRIX = { {0, 0, 1, 1, 0, 1},
           {0, 0, 1, 1, 0, 1},
           {0, 1, 0, 1, 0, 1},
           {0, 0, 0, 0, 0, 0},
           {1, 0, 1, 1, 0, 1},
           {0, 0, 0, 1, 0, 0},
```

Output example:

Homer is the person with the id 3

Input Example

```
MATRIX = { {0, 0, 1, 1, 0, 1},
           {0, 0, 1, 1, 0, 1},
           {0, 1, 0, 1, 0, 1},
           {0, 1, 0, 0, 0, 0},
           {1, 0, 1, 1, 0, 1},
           {0, 0, 0, 1, 0, 0},
```

Output example:

Everyone has talked to at least one other person. Homer is not present

If you have any questions, please e-mail: oanca.a.m@gmail.com

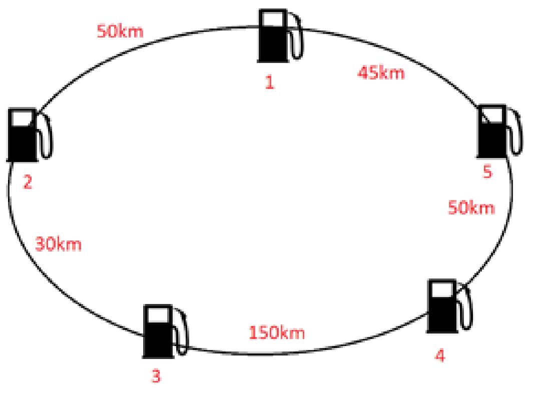
Ex 2 - Queue (3.5p)

During the summer holiday, you want to go on a tour around the country with your car. Obviously, you will end your journey in the place where you started it. You know the average fuel consumption of your car, denoted with FC (fuel consumption, a constant that is written in any car's specifications sheet)

There are N petrol stations in the country. Petrol stations are numbered from 0 to N.

For each petrol station, you know:

- the distance from that petrol station to the next petrol station in order (example : distance from 1 to 2, from 2 to 3, from 3 to 4, etc)
- the max amount of petrol that you can buy from that particular petrol station



Initially, your fuel tank is empty, and consider that its capacity is unlimited. You can start the journey at any of the petrol stations. Calculate the first point from where you will be able to complete the journey. Consider that because you want to be sure that you never run out of fuel, you will stop at each of the petrol stations.

Hint: In order to find out how many kilometres you can run with a certain amount of fuel, you can apply the following formula:

D = (100*F) / FC

where:

- F = fuel quantity (L)
- FC = fuel consumption (L/100km)
- D = max distance (KM)

Example: for a fuel consumption of 5L/100km, and a quantity of 2 litres you can travel 40 km

Input Format

The first line will contain the values of N, and FC.

The next N lines will contain a pair of integers each, i.e. the amount of petrol that petrol pump will give and the distance between that petrol pump and the next petrol pump.

Output Format

An integer which will be the smallest index of the petrol pump from which we can start the tour.

Sample Input

```
3 5

1 25

10 15

3 20
```

Sample Output

```
1
```

Explanation

We can start the tour from the second petrol pump:

3 5-> 3 stations, 5L/100KM consumption

25 1-> station index 0

15 3 -> station index 1

20 2 -> station index 2

If we start from station 0

Max distance we can travel = 20 km ->we cannot reach station 1 -> we can't start from station 0

If we start from station 1

Max distance we can travel = 50 km ->we can travel to station 2, where we add another 3 litres, meaning that we remain with (3+2) – (5*25/100) = 5-1.25 = 3.75 L

Now, from station 2, we travel to station 0, and after leaving it, we will still have enough fuel to go back to station 1, from where we started.

Result : we can start out journey from station 1

If you have any questions, please e-mail: alexandru.hang99@gmail.com

Ex 3 - Lists (3.5p)

Requirement

Let `N` be a natural number and `A` a vector. Initially, `A [i] = i` for all `i = 1, 2, ...,N`. Two types of operations can be applied to this vector:

`I St Dr`

Invert the subsequence of the vector that starts at position `St` and ends at position `Dr(1<=St<=Dr<=N)` For example, consider `N = 10`. Applying `I 2 7` will obtain the vector `1 7 6 5 4 3 2 8 9 10`

`S St Dr`

Display the sum of the elements in the subsequent sequence that starts at position `St` and ends at position `Dr` (`1<=St<=Dr<=N`). For example, consider `N = 10`. Applying `S 2 7` displays `27`.

Input data

The file `input.in` contains on the first line two natural numbers separated by spaces `N` and `K` (where `N` is the size of the vector and `K` is the number of operations to be performed). On the following `M` lines are written `K` operations, one operation on each line. These operations are written in the order in which they must be performed. An operation is described by a character (`I` or `S`) followed by space, then by the index `St` (`1<=St<=N`), then by space, then by the index `Dr` (`St<=Dr<=N`).

Output data

Thefile `output.out` will contain as many lines as type 2 operations (write operations) are in the input file.

Example

input.in	**output.out**	**Explanation**

`15 4`
`S 2 11`
`I 10 15`
`I 1 10`
`S 5 10`
	`65`
	`21`
		Initially the vector is `1 2 3 4 5 6 7 8 9 10 11 12 13 14 15`
		
		`S2 11` displays the sum of the elements from position 2 to position 11 including: `2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 = 65`
		
		`I 10 15` invert the subsequent elements from position 10 to position 15. The vector obtained is:
		
		`1 2 3 4 5 6 7 8 9 15 14 13 12 11 10`
		
		`I 1 10` invert the subsequent elements from position 1 to position 10. The vector obtained is: `15 9 8 7 6 5 4 3 2 1 14 13 12 11 10`
		
		`S 5 10` show the sum items from position 5 to position 10 including: `6 + 5 + 4 + 3 + 2 + 1 = 21`

Indications

We will use a **Doubly Linked List** in which we will retain the intervals obtained after the reversal operations. Initially the list consists of a single node containing the interval `[1, N]`.

For each inversion operation `I X Y`:
- determine the element in position `X`
- partition the interval in which the element is located at position `X`, in two intervals
- determine the element at position `Y`
- partition the interval in which the element is located at position `Y`, in two intervals
- traverse the nodes in the list between positions `X` and `Y` and invert the corresponding intervals

To calculate the sum of the elements `S X Y` it is sufficient to traverse all the nodes in the list corresponding to the intervals between the interval in position `X` and the interval in position `Y`, we will determine the sum of the elements in each interval and the sums are summed.

In the intervals where positions `X` and `Y` are, only the sum up to pos `X` and `Y` must be calculated.

Time complexity: O (K * K)
Memory: O (K)

Restrictions

- $1 <= K <= 5000$
- $0 < N <= 10^9$
- There is at least one write operation in the input file.

If you have any questions, please e-mail: dianascurtu14@gmail.com