South African Computer Olympiad
Final Round 2010
Day 2

Overview

Problem	**coaster**	**track**	**haunted**	**ppp**
Source	coaster.java coaster.py coaster.c coaster.cpp	track.java track.py track.c track.cpp	haunted.java haunted.py haunted.c haunted.cpp	ppp.java ppp.py ppp.c ppp.cpp
Input file	stdin	stdin	stdin	stdin
Output file	stdout	stdout	stdout	stdout
Time limit	1 second	0.2 seconds	0.5 seconds	1 second
Memory limit	128MiB	128MiB	128MiB	128MiB
Number of tests	10	10	10	10
Points per test	10	10	10	10
Detailed feedback	Yes	Yes	No	No
Total points	**100**	**100**	**100**	**100**

The maximum total score is 400 points.

Rollercoaster Rejection

Introduction

The Eight-Planets Individuon 9000 rollercoaster is unlike any other: it caters for everyone – from the unseasoned newcomer to the hardened thrill-seeker. Each of the rollercoaster's single-seater cars has carefully tuned shock absorbers and brakes to give a different level of intensity to the ride.

Riders each have a maximum level of intensity they will enjoy. They would like the ride as close to their personal intensity limit as possible, but not over it.

Task

The rollercoaster cars are numbered from 1 to K (in order from front to back), and there are N people in the queue. The car at the front has gives the most intense, and the ride intensity decreases the further back the car.

In order to enjoy the ride as much as possible, each rider in turn will go to the car that will give them the maximum enjoyment: the most intense one they can handle. If that car is empty, they will sit in it; otherwise, they will go to the car behind it. If they reach the end of the rollercoaster without finding an empty car, they will go home rather than risk a heart attack.

Of course, the park's management doesn't want guests leaving unsatisfied. They would like you to write a program to predict which guests would end up going home so that they can suggest another ride.

Example

Suppose that there are five cars in the rollercoaster and four people in the queue: first Alice, then Bob, followed by Chris and Dianne. Alice and Bob can handle cars 4–5, but Chris will only sit in car 5. Dianne can handle any level of intensity.

In this case, Chris would end up going home:

- Alice sits in car 4.
- Bob goes to car 4, finds it full, and goes to sit in car 5.
- Chris goes to car 5 and finds it full, but won't sit in cars 1–3.
- Dianne sits in car 1.

Input (stdin)

The first line of input contains two space-separated integers, K and N. The next N lines represent the people in the queue, in order: each line contains an integer from 1 to K, the most intense car the corresponding person would enjoy.

Sample input

```
5 4
4
4
5
1
```

Output (stdout)

Output the number of each person who will go home (numbering the people in the queue from 1 to N), with one number on each line, in order. If everyone will find a satisfactory seat, do not output anything.

Sample output

```
3
```

Constraints

- $1 \leq K \leq 100$
- $1 \leq N \leq 100$

Time limit

1 second. Python multiplier: 10. Java multiplier: 2.

Detailed feedback

Detailed feedback is enabled for this problem.

Scoring

A correct solution will score 100% while an incorrect solution will score 0%.

Keeping Track

Introduction

Any large theme park needs a train system to help guests get around. Track & Train, Inc. sells prefabricated train tracks, but their ordering process is a little unusual.

Task

In the Track & Train catalogue are N straight sections of track, numbered from 1 to N. The ordering process only allows you to specify two numbers, a and b, to buy all the track pieces numbered between a and b, inclusive.

You must write a program to determine whether it is possible to order an exact total length of track in a single order and, if it is, output a and b. If there is more than one way to do so, output any of them.

Example

Suppose the following track pieces are available:

1. 5 m
2. 4 m
3. 2 m
4. 3 m
5. 7 m

Then it is possible to order 9 metres of track with pieces 1 to 2 or pieces 2 to 4. It is not possible to get 8 metres in a single order.

Input (stdin)

The first line of input contains two space-separated integers, the desired length of track (in metres) and N. The next N lines each contain a single integer, the respective lengths (in metres) of track sections 1 to N.

Sample input

```
9 5
5
4
2
3
7
```

Output (stdout)

The output consists of a single line. If it is impossible to make a single order for the desired length of track then the line should contain the word “**IMPOSSIBLE**”; otherwise, it should consist of two space-separated integers, a and b.

Sample output

```
1 2
```

Constraints

- $1 \leq N \leq 100\,000$
- $1 \leq$ length of each track piece $\leq 10\,000$
- $1 \leq$ desired track length $\leq 1\,000\,000\,000$

Additionally, in at least 40% of the test cases:

- $1 \leq N \leq 1\,000$

Additionally, in at least 20% of the test cases:

- $1 \leq N \leq 100$

Time limit

0.2 seconds. Python multiplier: 10. Java multiplier: 2.

Detailed feedback

Detailed feedback is enabled for this problem.

Scoring

A correct solution will score 100% while an incorrect solution will score 0%.

Haunted House II

Introduction

Having escaped from the Bentopian Scare House, Bruce has bought a ticket for Carl to visit another haunted house. This haunted house consists of a series of rooms connected by doorways. The rooms are connected in a structure called an unrooted tree. This means that:

- there is always a way to get from one room to another (possibly through other rooms),
- there is **only one** way to get from one room to another room if you only visit each room at most once.

Carl is only allowed to visit a room at most once, since Bruce bought Carl a cheap ticket.

Task

Bruce wants Carl to visit as many rooms as possible, since this will increase the number of times Carl is scared. Given the layout of the haunted house, you must determine the maximum number of rooms Carl can visit in one go.

Carl can start and end his visit from any room in the haunted house, but he can only visit each room at most once.

Example

The example layout is shown in Figure 1. Carl can visit five rooms. He starts in room 7, walks to room 2 through rooms 3, 5 and 6. This is the maximum number of rooms Carl can visit without visiting a room twice. There is another way to visit five rooms: start in room 4 and then walk to room 7 through rooms 6, 5 and 3. Carl can also visit five rooms by following these routes in reverse order.

Input (stdin)

The first line of input contains a single integer N, the number of rooms in the haunted house. The following $N-1$ lines each contain two space-separated integer a_i, b_i; which tells you there is a doorway between room a_i and b_i. The doorway can be used to enter a_i from b_i, or to enter b_i from a_i. Rooms are labelled from 1 to N.

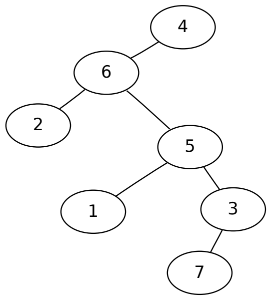

Figure 1: The example layout. Rooms are represented by circled numbers, while doors are lines between them.

Sample input

```
7
2 6
6 5
3 7
4 6
1 5
5 3
```

Output (stdout)

The output must contain a single integer L, the maximum number of rooms Carl can visit in one go.

Sample output

```
5
```

Constraints

- $1 \leq N \leq 50\,000$
- $1 \leq a, b \leq N$

Additionally, in 70% of the test cases, $N \leq 1\,000$. Additionally, in 40% of the test cases, $N \leq 10$.

Time limit

0.5 seconds. Python multiplier: 10. Java multiplier: 2.

Scoring

A correct solution will score 100% while an incorrect solution will score 0%.

Programming Pollution Probes

Introduction

Dangerous pollutants have recently been detected in the theme park's River Rafting ride. The ride gets its water from a nearby river, which is being polluted by one of the many factories along its banks. The Environment Protection Force (EPF) wants to find out *which* factory – and they want your help to speed up the process.

Task

There are N factories evenly spaced along the river, with the first one at the river mouth. All water from the polluting factory to the mouth is polluted.

The EPF has a pollution probe mounted on a boat, which starts at the mouth. The boat takes T minutes to travel from one factory to the next, and the probe takes S minutes to sense whether the water at its location is polluted. It cannot operate while the boat is moving.

The EPF needs to know which factories they should probe, and when, to minimize the time needed to find the polluting factory. Of course, they may be lucky and find the factory with the first probe, but they cannot rely on this in their planning. Therefore your program should find a plan to minimize the time to find the polluting factory in the worst case.

Example

Suppose there are three factories, with $S = 1$ and $T = 2$. We travel from factory 1 to factory 2, where we use the probe. If the water is unpolluted we conclude that factory 1 is polluting (in this case we take 3 minutes). Otherwise, the water is polluted, so we must move to factory 3 and use the probe to determine whether factory 2 or 3 is polluting (in this case we take 6 minutes).

Therefore this plan takes 6 minutes in the worst case. There is no better plan.

Input (stdin)

The input consists of a single line containing three space-separated integers: N, S, and T.

Sample input

```
3 1 2
```

Output (stdout)

The output consists of two lines, each containing a single integer. The first is the minimum worst-case time to find the polluting factory; the second is the first factory (numbering them from 1, at the river mouth, to N) that should be probed. If the minimum worst-case time can be obtained in more than one way, give any factory that could be probed first.

Sample output

```
6
2
```

Constraints

- $2 \leq N \leq 10\,000$
- $0 \leq S, T \leq 1\,000$

Additionally, in 60% of the test cases:

- $N \leq 1\,000$

Additionally, in 40% of the test cases:

- $N \leq 100$

Time limit

1 second. Python multiplier: 10. Java multiplier: 2.

Scoring

A correct solution will score 100%. A solution with the correct worst-case time but incorrect factory will score 60%. A solution with the incorrect worst-case time will score 0%.