Australian Informatics Olympiad
Thursday 6 September, 2012

Senior Paper

Duration: 3 hours

3 questions
All questions should be attempted

Problem 1
Posters

Input File: *postin.txt*
Output File: *postout.txt*

Time Limit: 1 second

You run a poster advertisement company. Your company is quite small: all it owns is a rectangular wall in the city. Advertisers pay to put up their poster on your wall at some time at some position along the wall. These posters may have different widths, but are all exactly the same height as the wall. When a poster is put up, it may cover some of the posters already on the wall.

You have a log of every single poster put on your wall: their distance from the left end, their width, and the time they were put up.

Since people always walk from the left to the right of your wall, and we all know advertisement posters are only effective when they are completely uncovered (e.g. a burger picture will only be mouthwatering if you see the whole burger), you would like to determine the leftmost fully visible poster.

Input

The first line will contain a single integer N, the number of posters that have been put up. The next N lines will contain descriptions of the posters in chronological order (from earliest to most recent). The ith of these lines will contain two integers x_i and w_i, which indicate the distance from the left end and the width of poster i respectively. (All units are in metres.)

Output

Output should consist of a single integer: the index i of the leftmost, completely visible poster (where the index of the first poster is 1).

Sample Input 1

dist from left

```
  4
3 1 5   width
  3 7
1 2 6
  8 9   fully shown
```

Sample Output 1

```
3
```

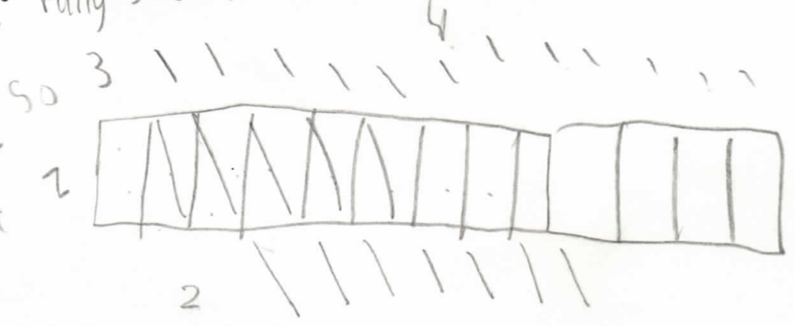
Sample Input 2

```
  7
1 40 50   covered
2 30 45   covered
3 1 10    ~~not sh~~ uncovered
4 20 30   covered
5 1 30    covered
6 5 15    covered
7 10 40   fully shown
  x  w
```

Sample Output 2

```
7
```

covered area . .
$x_{lowest} \rightarrow (x_i + w_i) - 1)$ largest



x_i is greater than w last
or
$w_i + x_i - 1$
less than x last

Explanation

In the first sample case, the first and second posters are both partially covered by the third poster. While the fourth poster is completely visible, it is further to the right of the third poster and does not overlap it, so the third poster is the leftmost completely visible poster.

In the second sample case, the first poster is partially covered by the second, the second is partially covered by the seventh, the third and fourth are covered by the fifth, and the sixth is partially covered by the seventh. The seventh poster is the only completely visible poster.

Constraints

To evaluate your solution, the judges will run your program against several different input files. All of these files will adhere to the following bounds:

- $1 \leq x_i \leq 1,000,000$
- $1 \leq w_i \leq 10,000$
- $1 \leq N \leq 100,000$

As some of the test cases will be quite large, you may need to think about how well your solution scales for larger input values. However, not all the cases will be large. In particular:

- For 60% of cases, $1 \leq N \leq 200$.

Scoring

The score for each input scenario will be 100% if the correct answer is written to the output file, and 0% otherwise.

Problem 2
Cabinet Shuffle

Input File: *shufflein.txt*
Output File: *shuffleout.txt*

Time Limit: 1 second

The polls are looking grim for the government of Absurdistan. Leadership speculation and high-profile scandals dominate popular current affairs shows *Tomorrow This Morning* and *An Antiquated Event*. In order to radically change public perceptions, the leaders plan to remove a ministry position and blame all the problems of the day on the ousted minister. At the same time, the other cabinet positions will be shuffled so as to portray a fresh, new face of government.

Naturally, the ministers can not agree between themselves who will be blamed and expelled, nor can they agree who will take which remaining ministry positions (including the position of Prime Minister). They decide to play a fair game of Musical Chairs to the tune of *Party Rock Anthem* in order to resolve these disputes.

There are K ministry positions available, each represented by a physical seat at a point around a circle. The $K+1$ ministers are also initially standing at points around the circle. Points on the circle are labelled clockwise from 1 to N, such that point 1 immediately follows point N. No two ministers will be initially standing at the same point, and no two chairs will be at the same point.

Each second, all the ministers who are still standing do the following (simultaneously):

- If the minister is standing at the same point as an empty chair, the minister will sit down in it.
- Otherwise, the minister will step one place clockwise around the circle to the next point. If the minister was previously at point i (with $i < N$), the minister will now be at point $i+1$. If the minister was previously at point N, the minister will now be at point 1.

Since there are $K+1$ ministers, eventually all K seats will be taken and the one minister remaining without a seat will be booted out and shamed by the media. Furthermore, the minister sitting in the first seat in the circle will have the place of Prime Minister. (The 'first' seat in the circle is defined as the first seat clockwise from point 1.)

Your task is to determine who will be Prime Minister and who will be expelled from cabinet after the reshuffle. Note that your program can score half of the available marks for correctly answering only one of these questions, and will score full marks for correctly answering both.

points on circle

Input

Your program should read from the file `shufflein.txt`.

"N K"

- The first line of input will contain two space-separated integers N and K.
- The second line of input will contain K space-separated integers representing the points at which there are chairs, in increasing order. Hence, the first chair in this list will be the Prime Minister's. seats
points of chair
- The third line of input will contain $K+1$ space-separated integers representing the points of ministers, in increasing order. The ministers are numbered from 1 to $K+1$ by their position in this list.

+3
points of ministers

K being 4

1 2 3 4 5

N K

Output

Your program should write to the file `shuffleout.txt`. Your output file should contain two lines.

- The first line should contain one integer: the number of the minister who is in the first seat at the end of the game.
- The second line should contain one integer: the number of the minister who is left with no seat at the end of the game.

Sample Input

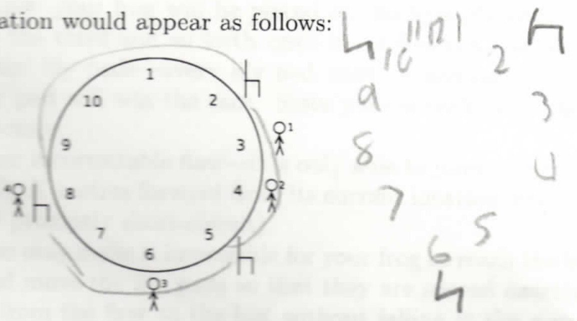
```
10 3
2 5 8      points of chairs
3 4 6 8    points of minister
```

Sample Output

```
3   pm
1   \ Spare
```

Explanation

For illustration, the initial configuration would appear as follows:



In the first second, the fourth minister (at point 8) will immediately sit down on the chair beneath her. The other three ministers will move one place around the circle. In the next second, the second minister (now in position 5) will sit down on the second seat. The first and third ministers will continue walking around the circle until the third minister reaches position 2 and sits down, leaving the first minister with no chair to sit in.

Constraints

To evaluate your solution, the judges will run your program against several different input files. All of these files will adhere to the following bounds:

- $1 \leq N \leq 1,000,000$
- $1 \leq K \leq 100,000$
- All chair and minister positions will be integers between 1 and N inclusive. No two chairs will have the same position and no two ministers will have the same initial position.

As some of the test cases will be quite large, you may need to think about how well your solution scales for larger input values. However, not all the cases will be large. In particular:

- For 30% of the available marks, $1 \leq N \leq 1,000$.

Scoring

Your score for each test case will be 100% if both lines are correct, 50% if exactly one is correct, and 0% otherwise.

Problem 3
Awesome Frog

Input File: *frogin.txt*
Output File: *frogout.txt*

Time Limit: 1 second

The inaugural International Olympiad in Frogleaping is being held in Australia in 2013 and you are determined to win. While you want nothing to do with such slimy, jumpy creatures, you plan to enter a frog-like robot that you know will be faster than all the other organic entrants.

The IOF takes place in a large pond in which there is a sequence of lily pads arranged in a long line. The rules of the race are simple: your frog will be placed on the first lily pad, then it must jump to the second lily pad, then the third and so forth until it reaches the last lily pad in the course. Note that you can not 'skip' lily pads—every lily pad must be jumped on exactly once. The first frog to reach the last lily pad will win the race. Since your robotic frog has super-frog speed, you are confident in your victory.

However, your frog has one minor incorrectable flaw—it is only able to jump one fixed distance. Specifically, it can only jump *exactly* K metres forward from its current location, even if this lands the frog in the water (where it will promptly short-circuit).

Since the initial lily pad positions may make it impossible for your frog to reach the last lily pad, you plan to create a distraction and move the lily pads so that they are spaced exactly K metres apart, enabling your frog to jump from the first to the last without falling in the water. Shifting a lily pad by one metre will take you one second, and the longer you spend stealthily moving lily pads, the more likely that the IOF judges will notice and disqualify you from the competition.

Given the initial distances between the lily pads in the course, you must write a program to compute the minimum time you will have to spend shifting lily pads such that all pairs of consecutive lily pads are exactly K metres apart. You can assume that the pond is sufficiently long so that the first lily pad can be moved any distance back, and the last lily pad can be moved any distance forward.

Input

Your program should read from the file `frogin.txt`. The first line of this file will consist of two space-separated integers N and K. The following $N-1$ lines will contain the initial distances between consecutive pairs of lily pads. Specifically, the ith line will contain one integer representing the distance between the ith and $(i+1)$th lily pad.

Output

Your program should write to the file `frogout.txt`. Your output file should consist of one line containing one integer: the minimum total time spent shifting lily pads so as to ensure your victory.

Sample Input

```
5 6
8
3
6
4
```

Sample Output

```
6
```

Explanation

In this example, if we shift the first lily pad back by 1 metre and the second lily pad back by 3 metres, there will be 6 metres between the first and second and 6 metres between the second and third lily pads. The third and fourth are already 6 metres apart, but we must move the fifth lily pad forward by 2 metres to put it 6 metres after the fourth.

The total time taken is $1 + 3 + 2 = 6$ seconds.

Constraints

To evaluate your solution, the judges will run your program against several different input files. All of these files will adhere to the following bounds:

- $1 \leq K \leq 20,000$
- $2 \leq N \leq 100,000$
- The initial distance between each consecutive pair of lily pads will be between 1 and $20,000$ metres.
- It is guaranteed that the minimum time you will have to spend shifting lily pads will be at most $2^{31} - 1$.

As some of the test cases will be quite large, you may need to think about how well your solution scales for larger input values. However, not all the cases will be large. In particular:

- For 30% of cases, $N \leq 100$, and distances between consecutive pairs of lily pads will be at most $1,000$ metres.

Scoring

The score for each input scenario will be 100% if the correct answer is written to the output file, and 0% otherwise.