South African Computer Olympiad
Camp 1, 2009
Day 1

Overview

Author(s)	Julian Kenwood	Marco Gallotta	Max Rabkin
Problem	**broken**	**wrapping**	**washing**
Source	broken.c broken.cpp	wrapping.c wrapping.cpp	washing.c washing.cpp
Input file	stdin	stdin	stdin
Output file	stdout	stdout	stdout
Time limit	1 second	1 second	1 second
Number of tests	10	10	10
Points per test	10	10	10
Detailed feedback	Yes	No	Yes
Total points	**100**	**100**	**100**

The maximum total score is 300 points.

Broken Compiler

Julian Kenwood

Introduction

Bruce and Carl have recently left university to start their own software company. Unfortunately, they only had enough money to buy a broken compiler for their projects. They have just completed the code of their first project for the Vocational Guidance Counselor's office, but their compiler is just too broken to compile their project. You must help Bruce and Carl compile their project.

Task

The project has N modules numbered 1 to N. Each module may have several dependencies. Before a module can be compiled, all of its dependencies must have already been compiled.

The compiler is not only broken, but also malicious. It will only accept compiling of the project's modules in the lexicographically smallest possible ordering. Your job is to find this ordering.

Example

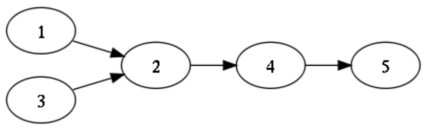

Figure 1: *Example project.*

Assume Bruce and Carl's project is broken up into 5 modules with 4 dependencies. Modules 1 and 3 each depend on module 2. Module 2 depends on module 4, while module 4 depends on module 5.

From figure 1, we can see that we must compile modules 5, 4 and 2 before modules 1 and 3. This gives us two possible orderings: both 5 4 2 1 3 and 5 4 2 3 1 will compile the entire project. Ordering 5 4 2 1 3 is lexicographically the smallest (1 comes before 3) and is therefore the answer.

Input (stdin)

The first line of input contains two space-separated integers, N and D: the number of modules and number of dependencies respectively. The next D lines each contain two space-separated integers, A_i and B_i: indicating that module A_i depends on (and must be compiled *after*) module B_i.

Sample input

```
5 4
1 2
3 2
2 4
4 5
```

Output (stdout)

Output N integers one per line, representing the lexicographically smallest ordering. If no such ordering exists, output only -1.

Sample output

```
5
4
2
1
3
```

Constraints

- $1 \leq N \leq 100\,000$
- $1 \leq D \leq 200\,000$
- No two dependencies will be equal

Additionally, in 50% of the test cases:

- $1 \leq N \leq 10\,000$
- $1 \leq D \leq 20\,000$

Time limit

1 second.

Detailed feedback

Detailed feedback is enabled for this problem.

Scoring

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

Wrapping the Rock

Marco Gallotta

Introduction

Keegan, the infamous local bergie, has had enough of watching everyone celebrate Christmas and has decided this is the year he'll join in the fun. He has chosen the perfect present for his Uncle Lucky — the largest rock he can find. Unfortunately though, he's made a terrible job of wrapping up the rock and has left many sections uncovered.

Task

Keegan can purchase sheets of wrapping paper large enough to cover K millimeters of the rock's circumference. He cannot afford scissors so he cannot cut up the sheets into smaller pieces. Help him determine the minimum number of sheets he needs to purchase in order to cover the uncovered sections. A single sheet is **not** long enough to cover two sections that are exactly K millimeters apart. Covering a section with more than one sheet is acceptable, but every section must be covered. The rock is shaped such that the uncovered sections form a ring around the rock, allowing sections neighbouring on either side of the section at location 0 to be covered by a single sheet, if wide enough of course.

Example

Consider a rock $R = 10$ millimeters in circumference, with uncovered sections at 0, 9, 4 and 6 millimeters (Keegan randomly chooses a reference point which he labels position 0). The sheets can each cover $K = 2$ millimeters of the rock's circumference. The sections at 0 and 9 millimeters can be covered with a single sheet, since the rock is circular. The other two sections require two separate sheets, giving a total of three sheets required to wrap the rock.

Input (stdin)

The first line of input contains three space-separated integers: R, the circumference of the rock; N, the number of uncovered sections; and K, the length (in millimeters) of the rock's circumference that a sheet of wrapping paper can cover. The next N lines each contain a single integer: the location L_i of an uncovered section in millimeters. The locations are provided in sorted order.

Sample input

```
10 4 2
0
4
6
9
```

Output (stdout)

Output a single integer: the minimum number of sheets required to wrap the rock.

Sample output

```
3
```

Constraints

- $1 \leq R \leq 2^{31}$
- $0 \leq N \leq 1\,500\,000$
- $1 \leq K \leq R$
- $0 \leq L_i < L_j < R$ for all $i < j$

Additionally, in 80% of the test cases:

- $0 \leq N \leq 750\,000$

Additionally, in 40% of the test cases:

- $N \leq 80\,000$
- $1 \leq K \leq 100$

Scoring

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

Washing Line

Max Rabkin

Introduction

Fred the manic storekeeper is doing his laundry, but has discovered that his washing line is rather weak. He wants to find the segment of the washing line with the heaviest load of clothes, so that he can reinforce that segment to stop the line breaking.

For some reason, Bruce's laundry got mixed in with Fred's, so some of the laundry items have negative weights.

Because of his mania, Fred doesn't put the clothes up in order. From time to time, Fred will move the reinforcement to a different segment to take the new clothes into account; if no segment has positive weight, he won't bother to reinforce any segment.

Task

Given the sequence of weights on the line, and any updates, find the largest sum of weights of a consecutive subsequence.

Example

Say the original sequence of weights is $1, 2, -1$. Then the heaviest subsequence is $1, 2$, with weight 3. If an item of weight 4 is inserted at the end, then the heaviest subsequence will be the sequence itself, with weight 6.

Input/Output

Your program must read commands from standard input. The three commands are:

I p w Insert an item of weight w at position p. The position *before* the current first item is $p = 0$, and the position after the kth item is $p = k$. If there are n items on the line, then $0 \leq p \leq n$.

Q Query the heaviest subsequence. Your program must output the total weight of the current heaviest subsequence followed by a newline, and flush the output. If the heaviest subsequence is the empty sequence, output 0.

X Exit. The interaction is finished, and your program must exit.

The first command will always be an insertion.

Input	Output
I 0 1	
I 1 2	
I 2 -1	
Q	3
I 3 4	
Q	6
X	

Constraints

- There will be at most 100 000 commands.
- $-1000 \leq w \leq 1000$

Additionally, in 40% of the test cases: there will be at most 5 000 commands.

Additionally, in 20% of the test cases: there will be at most 500 commands.

Time limit

1 second.

Detailed feedback

Detailed feedback is enabled for this problem.

Scoring

If your interaction with the evaluator is in the wrong format, you will receive zero. Otherwise, let Q be the number of queries, and C the number of queries answered correctly. If $Q = C$ you will receive 10 points; otherwise you will receive $\lfloor \frac{Q}{C} \times 5 \rfloor$ points.