2nd Annual CSSA Programming Contest
February 2, 2019

General Instructions:

1. Submit solutions using the PC^2 software.
2. All input should be read from standard input.
3. All output should be written to standard output.
4. Each problem has a twenty (20) second time limit for CPU time when executed on the judging data.
5. Problems descriptions are not ordered by level of difficulty.

**Sponsors**

norima CONSULTING

TELENIUM®

SKIP THE DISHES

## Example Problem: Best Stock Sale **(Do not submit)**

You're an investor on wall street, and you believe that $SNAP is seriously undervalued right now. In the past you've believed the same, but have been wrong most of the time. During a period of reflection, you decide to determine when the best time to buy and sell one unit of $SNAP would have been.

Given a list of historical stock prices for $SNAP, determine the maximum amount of profit that could have been earned by a single trade. You are allowed to buy exactly once, then sell exactly once at some point in the future. Profit is equal to selling price minus buying price.

For example, given stock prices [7, 1, 5, 3, 6, 4], the best trade is to buy at $1 and sell at $6, giving a total profit of $5.

Sometimes not buying at all would be a good idea. Given stock prices [7, 6, 4, 3, 1], you should not trade at all, giving a total profit of $0.

### Input

The first line of input contains an integer $2 \le N \le 10{,}000$ giving the number of historical prices you have access to. The next line of input contains N integers. Each integer P represents the stock price at some time in the day, with $0 \le P \le 1{,}000$. The integers on this line are ordered by time of day.

### Output

Output the maximum profit from a single stock trade. If none of the trades produce profit, you should output 0.

| Sample Input | Sample Output |
|---|---|
| `6`<br>`7 1 5 3 6 4` | `5` |

| Sample Input | Sample Output |
|---|---|
| `5`<br>`7 6 4 3 1` | `0` |

## Example Solution

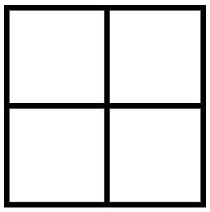
```java
import java.util.Scanner;

public class Solution {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        // read the input into an array
        int stockCount = input.nextInt();
        int[] prices = new int[stockCount];
        for (int i = 0; i < stockCount; i++) {
                prices[i] = input.nextInt();
        }

        // calculate the best possible trade
        int maxProfit = 0;
        int lowestPriceSeen = Integer.MAX_VALUE;
        for (int i = 0; i < prices.length; i++) {
                lowestPriceSeen = Math.min(lowestPriceSeen, prices[i]);
                maxProfit = Math.max(maxProfit, prices[i] - lowestPriceSeen);
        }

        System.out.println(maxProfit);
    }
}
```

The above solution uses Scanner's nextInt() to read integer input. Consult the standard documentation to read types other than integer.
https://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html

Other solutions would also be accepted for the above problem. For example, if you were to use a nested for-loop to try every possible stock combination you would still have a correct answer.

This example is given to illustrate the usage of Java's Scanner class.

## Problem A: How many squares?



One day in math class, your friend hands you a drawing of a 2x2 grid and asks you how many squares you see. After answering 4, your friend grins as they explain to your surprise that there are actually 5 squares. Your friend was counting the bounding square as well! In the 2x2 grid there are four 1x1 squares that you counted, and one 2x2 square bounding the grid.

Out of curiosity, you start to draw more grids to count their squares. Everything from 1x1 to 10x10, and even some rectangular grids like 4x5 and 5x10. Eventually you get tired of counting and decide to write a program to solve the problem for you.

Given an MxN grid, you want to create a program to calculate the number of squares in that grid.

### Input

The input consists of a single line, giving the width and height of a grid. Specifically, the line has two integers describing the width W and height H of the grid, with $0 \leq W, H \leq 1000$.

### Output

Output a single integer giving the total number of squares in the grid.

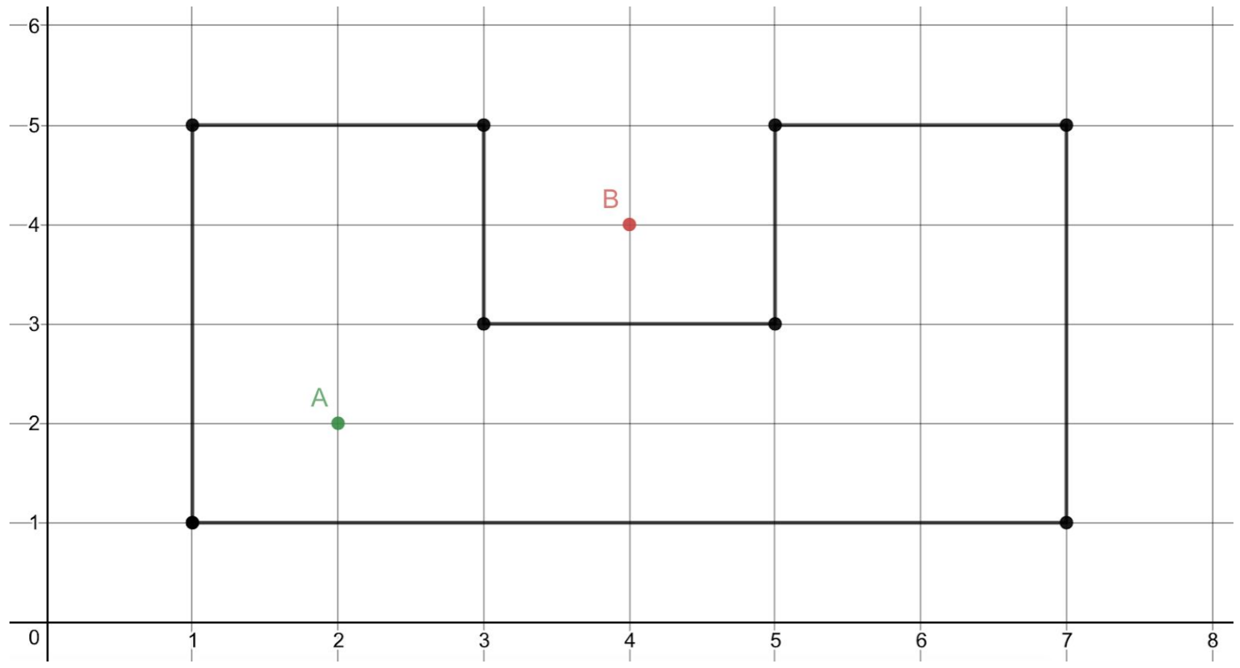
| Sample Input | Sample Output |
|---|---|
| `2 2` | `5` |

| Sample Input | Sample Output |
|---|---|
| `4 3` | `20` |

## Problem B: Geofencing API

You’re developing a ride-sharing application, and are tasked to develop the API for the application’s geofencing functionality. Each driver is assigned to a particular service area, and only responds to requests that are contained within the boundary of their service area. The boundary of their service area is called a geofence. When a driver receives a request from a rider, the geofencing API must be able to determine whether that rider is within the driver’s geofence.

The rider location is always an integer coordinate on the map, and will either be inside or outside the geofence (never on the fence). Geofences consist of only horizontal and vertical line segments, and have no openings on their perimeter. Every segment in the geofence starts and ends on integer coordinates. Consider the example below.



The above diagram contains a geofence, along with two rider locations labelled A and B. In this example, rider A is inside the geofence, and rider B is outside the geofence.

Given the description of a geofence and a number of rider locations, determine whether each rider is on the inside or outside of the geofence.

## Input

The input to the problem is a description of a geofence and a sequence of rider locations. The first line of input gives two integers, the number of segments in the geofence N, and the number of rider locations being considered K. The integers satisfy $4 \le N \le 10{,}000$ and $1 \le K \le 10{,}000$. The next N lines of input give one segment of the geofence per line.

Each segment is given as four integers $0 \le x_1, y_1, x_2, y_2 \le 10^9$, where $(x_1, y_1)$ and $(x_2, y_2)$ are the coordinates of the segment endpoints. The remaining K lines of input give the coordinates of a rider location. Each rider location is given as two integers $0 \le x_R, y_R \le 10^9$ where $(x_R, y_R)$ is the coordinate of the rider. It is guaranteed that $(x_R, y_R)$ is not on any segment of the geofence.

## Output

For each rider location, output “IN RANGE” or “OUT OF RANGE” to indicate whether that rider is on the inside or outside of the geofence. Each location’s output should be on its own line, and printed in the same order the locations were read in.

| Sample Input | Sample Output |
|---|---|
| <pre>8 2<br>1 1 1 5<br>1 5 3 5<br>3 5 3 3<br>3 3 5 3<br>5 3 5 5<br>5 5 7 5<br>7 5 7 1<br>7 1 1 1<br>2 2<br>4 4</pre> | <pre>IN RANGE<br>OUT OF RANGE</pre> |

## Problem C: Collatz Primes

The Collatz sequence is defined by starting with any positive integer as the first term. The next term in the sequence can be generated as a function of the current term. It is conjectured that any starting term will eventually reach 1 as part of the sequence.

Let X be the current term. The next term in the sequence is determined by the following rules:

1. If the X is even, the next term is X / 2.
2. If the X is odd, the next term is 3X + 1.

The number of steps to reach 1 can be counted by the number of times you generate a term before the current term is 1. We're concerned with the total number of steps in the Collatz sequence for prime numbers.

### Input

This problem does not take any input.

### Output

For the first 25 prime numbers (starting from 2), output the number of steps in the Collatz sequence to reach the term 1. Each number of steps should be on its own line.

Sample Output

```
1
7
5
16
```

*(remaining output not included)*

## Problem D: Task Management

You are a manager that has to delegate many incoming tasks, and thankfully you have good staff. You don't have the time to match the perfect task to the perfect worker. So you devise a simple system, workers come to you one at a time for tasks and can do any of the following:

- view the easiest task
- view the hardest task
- take the easiest task
- take the hardest task

Head office is ruthless though, so throughout the day you might also receive new tasks while you are trying to distribute tasks. These tasks will have to be considered in future requests from workers.

You also have to be careful about your system's efficiency, because workers don't like to wait and will need a quick response from you. You should not have to read the entire book of tasks every time a worker asks you for a task, as this would be too slow for them. Requests from head office should be similarly fast if you want to ensure you don't fall behind, though you might be able to get away with being a little bit slower here depending on how you built your system.

Given the description of requests from workers and head office, you must create a system that responds to requests from workers, and allows head office to add additional tasks.

### Input

Each line of input contains a single command to process. Commands start with a letter indicating the type of command, followed by any information required by the command separated by spaces. The format of the commands are as follows:

- `i` - Insert tasks. The remainder of the line contains at most 10 tasks to insert, each given by an integer $0 \le t \le 10^6$ representing the time that task takes to complete.
- `f` - Find tasks. The remainder of the line contains either "min" or "max", indicating whether you should find the minimum or maximum length task.
- `r` - Remove tasks. The remainder of the line contains either "min" or "max", indicating whether you should remove the minimum or maximum length task.

If you are asked to find or remove a task, there will always be at least one task waiting to be completed. There may be multiple tasks which have the same difficulty. You will receive at most 20,000 instructions. Input is finished when there are no more lines to read.

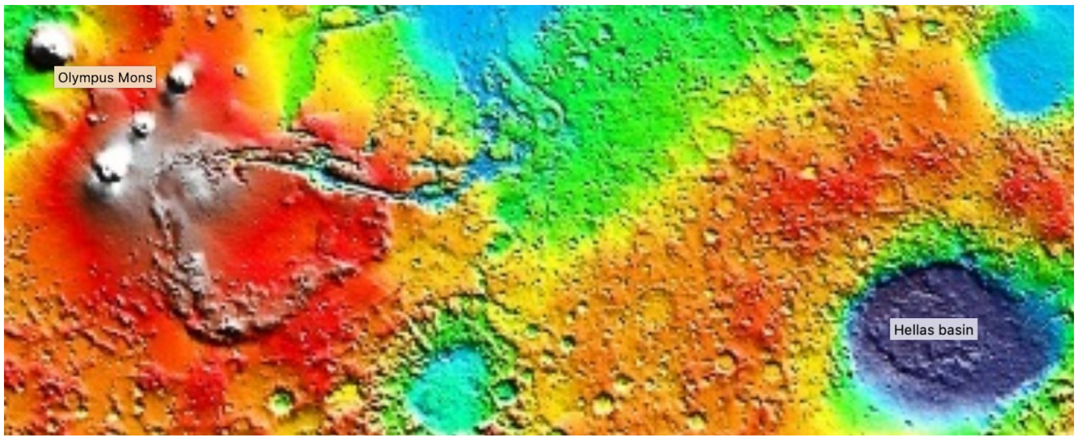
## Output

After processing each find command, you should output the difficulty of the task found. Similarly, after processing each remove command, you should output the difficulty of the task that was removed. Output from each command should be on its own line.

| Sample Input | Sample Output |
|---|---|
| `i 447 813 694 825 193 59 32 183 565 573`<br>`r min`<br>`f max`<br>`i 986`<br>`r max`<br>`f min` | `32`<br>`825`<br>`986`<br>`59` |

# Problem E: Givers and Takers

It’s the year 3019. After the initial colonization of Mars near Olympus Mons, humans expanded their settlements in a grid-like fashion across the highlands toward the Hellas impact basin. Each settlement developed at different rates over several hundred years, resulting in a large disparity between various settlements. Some developed settlements trying to increase tourism are called “givers”, and will pay you space bucks to visit. Other settlements needing funding for development are called “takers”, and will charge a tax in space bucks if you visit.



A topographic map of Mars, with Olympus Mons and the Hellas basin shown

You just graduated from university and are going on a backpacking trip across Mars starting at the most north-western settlement by Olympus Mons, and want to travel all the way to the most south-eastern settlement by the Hellas basin. Due to time constraints, your route must always head toward the goal by moving either south or east between settlements (never north or west). As you visit each settlement you will either gain space bucks from givers, or lose space bucks from takers.

Consider the grid shown to the right. This grid represents a 3x3 settlement map, with negative values indicating takers, and positive values indicating givers. For example, the top-left cell (where you start) is a settlement that will take 3 space bucks to visit. The path requiring the least amount of initial space bucks in this map is EAST EAST SOUTH SOUTH, requiring only 6 space bucks.

| | | |
|---|---|---|
| -3 | -2 | 3 |
| -6 | -12 | 1 |
| 8 | 17 | -5 |

Given a settlement map, determine the minimum amount of space bucks you need to bring at the beginning of your trip to ensure you can reach your destination without failing to pay any visiting fees.

## Input

The input to the problem is a grid of integers describing the settlement map. The value of each integer indicates the number of space bucks that a settlement will give or take from you.

The first line of input gives two integers describing the width W and height H of the settlement map, with 0 ≤ W, H ≤ 1000. The following H lines contain W integers each, with each integer X being in the range -1000 ≤ X ≤ 1000. These integers describe the grid.

## Output

Output a single integer giving the minimum number of space bucks needed to complete the trip..

| Sample Input | Sample Output |
|---|---|
| 3 3<br>-3 -2 3<br>-6 -12 1<br>8 17 -5 | 6 |

## Problem F: The Price Is Right

The popular game show “The Price Is Right” has various games about guessing the price of everyday and abnormal items. You’re competing in a game on the show where you are given a set of items with unique prices and need to guess which of two particular items is more expensive. The show’s host will pick which two items from the set you are comparing.

Often before coming on the show, contestants will try to memorize the price of items to gain an advantage. Unfortunately, you aren’t very good at remembering numbers, but you able to remember the price of some common items relative to each other. For example, you might know that item 1 is worth more than item 3, and that item 3 is worth more than item 5. Transitively, you know that item 1 is worth more than item 5.

Given your knowledge about relative prices between common items in the set, determine which of the two items given to you by the show’s host is more expensive, or whether you should guess because you don’t know for sure.

### Input

The problem input contains a description of which items you remember the relative weights for, followed by an indication of which items you must compare for the game.

The first line contains two integers N and M subject to $2 \le N \le 1000$ and $0 \le M \le 500{,}000$, where N is the number of items in the set, and M is the number of items you already know the relative weights for. Each of the next M lines contain two integers X and Y with $1 \le X, Y \le N$, indicating that you know item number X is more valuable than item number Y. The final line of input contains two integers P and Q subject to $1 \le P, Q \le N$, where P and Q are the two items given to you by the host.

### Output

If you know that item P is more valuable than item Q, you should output “P > Q”. Similarly, if you know that item Q is more valuable than item P, you should output “Q > P”. Finally, if you do not remember enough information to determine which item is more valuable, you should output “?”.

| Sample Input | Sample Output |
|---|---|
| 5 2<br>1 3<br>3 5<br>1 5 | Q > P |

# Problem G: Bad Counting

It's time to cash out after a long day working at the restaurant. Your boss takes the cash out of the register and calls you over to help count the money. You grab a pen and paper to keep count as your boss reads out the value of each coin and bill in cents.

Unfortunately, your boss frequently says the wrong number while reading out the contents of the register. Since these errors are so frequent, your boss tells you when to "zero" the most recent value by saying "zero". For example, your boss might say "5, 10, 5, 0" giving a total earning of 15 for the day.

Even more unfortunately, your boss might notice several errors. If this happens your boss will say "zero" multiple times to indicate that you need to zero out multiple recent values. For example, your boss might say "5, 10, 5, 0, 0, 25" giving a total earning of 30 for the day.

Given the entire sequence of readings from your boss, you must calculate the total amount of earnings for the day.

## Input

The first line of input contains an integer N indicating the number of values your boss will read out loud, and is bounded by $1 \le N \le 500{,}000$. The following N lines contain a single integer V giving the value of that item in the register, subject to $0 \le V \le 10{,}000$.

At any point in the sequence, there will be at least as many positive values as there are zero values. This implies that you will only be asked to zero out a recent value if one exists.

## Output

Output a single integer giving the total earnings for the day. The total earnings do not count any items that your boss had you zero out.

| Sample Input | Sample Output |
|---|---|
| 4<br>5<br>10<br>5<br>0 | 15 |

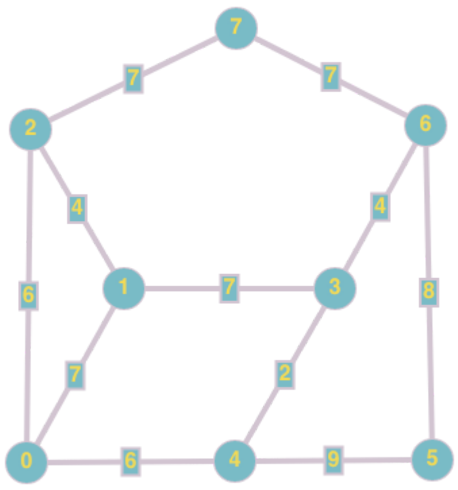
## Problem H: Building Bridges

Small communities frequently formed along rivers due to the access to transportation and resources provided by the river. Multiple communities might settle near each other along these rivers, but remain separated by the water until they decide to build bridges joining the communities.

River systems can be complicated, but we know the following properties:

- Communities are surrounded by 3 to 8 segments of the river system
- Two communities sharing a river segment are neighbours
- Communities that have an unshared river segment are neighbouring the outside world
- There is one community within each closed area of the river system
- There are no communities that exist outside the river system

The communities within the river system have grown so much that they're discussing a way to merge into one large city. Engineers in the communities observe the distances to cross each river segment into the neighbouring community, and use that to calculate the cost of building a bridge across any given segment.



Bridges can be built across river segments between neighbouring communities, or from communities to the outside world if they're neighbouring it. Your goal is to build bridges in order to obtain a land connection that enables access to every community by road. These bridges aren't cheap, so you want to do this while spending the least amount possible. Land connections are allowed to pass through the open world if desired.

The example shown in the diagram has 4 communities, 8 junctions, and 11 river segments that make up the river system. To connect the communities with minimum cost, bridges should be constructed across the river segments between junctions (1, 2) with cost 4, junctions (3, 4) with cost 2, and junctions (3, 6) with cost 4. The total cost to connect the communities with those bridges is 10, which is the minimum cost.

## Input

The first line of input contains an integer N giving the number of communities in the river system, subject to 0 ≤ N ≤ 100. The next N lines each contain a description of some community within the river system. The description consists of three parts:

- First, an integer S giving the number of river segments surrounding the community. We know from our properties above that 3 ≤ S ≤ 8
- Next, there will be S integers giving the labels for the river junctions that connect river segments surrounding the community, with labels being at most 1000
- Last, there will be another S integers giving the cost of building a bridge over each segment, with bridge costs being at most 5000

The descriptions of the river junction labels and bridge costs will be given in cyclical order. For example, the following description of a community

```
3 0 1 2 7 4 6
```

means there are three river junctions, and consequently three river segments. From the cyclic ordering, we know segment (0, 1) has cost 7, segment (1, 2) has cost 4, and segment (2, 0) has cost 6.

## Output

Output a single integer giving the minimum cost to connect the communities using bridges.

| Sample Input | Sample Output |
|---|---|
| 4<br>3 0 1 2 7 4 6<br>4 0 1 3 4 7 7 2 6<br>4 3 6 5 4 4 8 9 2<br>5 2 1 3 6 7 4 7 4 7 7 | 10 |

| Sample Input | Sample Output |
|---|---|
| 2<br>3 0 1 2 2 3 1<br>3 1 2 3 3 2 1 | 2 |