South African Computer Olympiad
Camp 2, 2008
Day 1

Overview

Author	**Nick Pilkington**	**Keegan Carruthers-Smith**	**Timothy Stranex**
Problem	**nearest**	**formula**	**mayor**
Source	nearest.java nearest.py nearest.c nearest.cpp nearest.pas nearest.hs	formula.java formula.py formula.c formula.cpp formula.pas formula.hs	mayor.java mayor.py mayor.c mayor.cpp mayor.pas mayor.hs
Input file	stdin	stdin	stdin
Output file	stdout	stdout	stdout
Time limit	2 seconds	1 second	5 seconds
Number of tests	10	10	10
Points per test	10	10	10
Total points	**100**	**100**	**100**

The maximum total score is 300 points.

Good Neighbours

Author

Nick Pilkington

Introduction

Bruce likes to visit his friends on the weekend. His friends are scattered around near where he lives and each lives at a unique location. Since Bruce does not have much time for visiting, he would like to make the most of his visits. Bruce wants to find the two friends that live closest to each other so that he can visit both of them in the same visit.

Having a lot on his mind, Bruce was rushed when making the list of friends' locations and thus may have specified some locations more than once. He trusts that you will be able to take this matter into account.

Task

Each of Bruces N friends live at some lattice coordinate (x_i, y_i). You will be given a description of the position of all Bruce's friends and need to compute the distance between the two closest friends. The distance metric used is the standard Euclidean distance in two dimensions:

$$\text{dist}(a, b) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}$$

Example

Suppose Bruce's friends are at the coordinates $(3, 0)$, $(3, 5)$, $(2, 4)$, $(4, 2)$ and $(0, 2)$ as illustrated in Figure 1. The two friend's at $(2, 4)$ and $(3, 5)$ are the closest at a distance of 1.41.

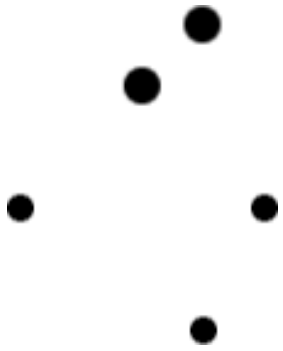

Figure 1: Example of the location of Bruce's friends. The two larger points represent the closest friends.

Input (stdin)

The first line contains a single integer, N. The next N lines each contain two space-separated integers x_i and y_i, the coordinates of a friend.

Sample input

```
5
3 0
3 5
2 4
4 2
0 2
```

Output (stdout)

Output a single number, the distance between the two closest friends rounded off to two decimal places.

Sample output

```
1.41
```

Constraints

- $2 \leq N \leq 100\,000$
- $0 \leq x_i, y_i \leq 1\,000\,000\,000$ for all i

50% constraints

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

Time limit

2 seconds.

Scoring

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

Where is The Formula?

Author

Keegan Carruthers-Smith

Introduction

Legend has it Bruce has a secret formula that transforms a problem statement into a coded solution. It is believed that this is what enables him to solve problems in under a minute as he is known to do.

Carl, believing this to be true, is working on trying to discover the secret formula. In order to know what to look out for he is running an experiment. Part of the experiment involves taking an equation and being able to evaluate it after changing a single variable each time.

Where is the formula? Where is it?

Task

Carl has asked for your help, but to stop you from learning his evil plot he has obfuscated the problem a bit.

You will be given an expression containing a number of variables. In the beginning, all variables default to zero. You are given M mutations, where each mutation changes the value of one variable. The variable is guaranteed to occur in the expression and will occur at most K times. Your task is to evaluate the expression after every mutation.

Postfix

Mathematical expressions are usually written in *infix* form, with the operations between operands. However, this notation is rather difficult for computers to parse, as precedence levels and brackets change the order of operations.

For that reason, *postfix* or *reverse Polish notation* is sometimes used. In this notation, both operands come before the operator. For example `(a + b) * c` becomes `a b + c *` and `a + b * c` becomes `a b c * +`. Thus, the order of operations depends only on their position in the expression, and brackets are not needed.

One way to evaluate postfix expressions is to use a stack. Scanning the expression from left to right, we look at each token. If it is an operand, we push it onto the stack. If it is an operator, we pop two operands off the stack and perform the operation on them, and push the answer onto the stack. When we reach the end, there is exactly one value on the stack: the value of the expression.

Example

Suppose we are given the expression provided in the sample input. Let us define $f(a, b, c)$ as the infix form of this expression:

$$f(a, b, c) = (((a + (3 * (a + c))) + b) + (a * b))$$

Plugging in the values given in the sample input we get:

$$\begin{aligned} f(1, 0, 0) &= 4 \\ f(1, 2, 0) &= 8 \\ f(2, 2, 0) &= 14 \\ f(0, 2, 0) &= 2 \\ f(0, 2, 10) &= 32 \\ f(1, 2, 10) &= 38 \end{aligned}$$

Input (stdin)

The first line contains 3 space-separated integers, N, M and K. The next line contains the postfix expression with N tokens. The next M lines each contain v_i and m_i. v_i is a string identifying a variable and m_i is its new integer value.

An expression token is 1 of 3 possibilities:

- *Operation* Either the string $*$ or the string $+$. Respectively representing multiplication and addition.
- *Constant* A string of digits, representing a non-negative integer.
- *Variable* A string of alphanumeric characters, with the exception of the first character, which must be an uppercase or lowercase letter.

Sample input

```
13 6 3
a 3 a c + * + b + a b * +
a 1
b 2
a 2
a 0
c 10
a 1
```

Output (stdout)

Output one line per mutation, the value of the expression after applying all mutations up until that one.

Sample output

```
4
8
14
2
32
38
```

Constraints

- $3 \leq N \leq 5\,000$
- $1 \leq M \leq 30\,000$
- $1 \leq K \leq 1\,000$

The result of the expression is guaranteed to always fit in a 64-bit unsigned integer.

50% constraints

- $3 \leq N \leq 2\,000$
- $1 \leq M \leq 15\,000$
- $1 \leq K \leq 200$

Time limit

1 second.

Scoring

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

Campaign planning

Author

Timothy Stranex

Introduction

Fred the manic store-keeper is running for mayor. He thinks he can win the election by playing on the loyalty of his store's customers. He wants to set up his campaign office in the area where most of his customers live. He has compiled a (long) list of the addresses of all his customers but needs your help to analyse it.

Task

His list contains customers numbered from 1 to N. The city is divided into a grid and the address of each customer is given as a point (x, y) on the grid. He has also made a list of M rectangles and wants you to find the customers who live within (boundary included) each rectangle. For each rectangle, you must output the number of customers living within it and the sum of their numbers modulo 1 000 000.

Example

Suppose there are four customers located at $(1, 1)$, $(2, 2)$, $(4, 3)$ and $(5, 4)$ and two rectangles with the first from $(1, 1)$ to $(5, 5)$ and the second from $(1, 2)$ to $(2, 2)$. The first rectangle contains all four customers so you would output 4 and $(1 + 2 + 3 + 4) \bmod 1\,000\,000 = 10$. The second rectangle contains only customer 2 so you would output 1 and $2 \bmod 1\,000\,000 = 2$.

Input (stdin)

The first line of the input contains two space-separated integers N and M. The next N lines each contain two space-separated integers x and y describing the addresses of the customers in order. The next M lines each contain four space-separated integers x_1, y_1, x_2 and y_2 describing a rectangle. The lower-left corner is given by (x_1, y_1) and the upper-right corner by (x_2, y_2). The rectangle includes its corner points and boundary.

Sample input

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

Output (stdout)

Output M lines containing the outputs for the rectangles in the same order as in the input. Each line should contain two space-separated integers. The first is the number of customers in the rectangle. The second is the sum of the numbers of these customers modulo 1 000 000.

Sample output

```
4 10
1 2
```

Constraints

- $1 \leq N \leq 10^5$
- $1 \leq M \leq 10^5$
- $1 \leq x, y \leq 10^9$
- The total number of customers found in all the rectangles will be $\leq 10^5$.
- No two customer addresses will have the same x or y coordinates.
- $x_1 \leq x_2$ and $y_1 \leq y_2$

50% constraints

- $1 \leq M \leq 1\,000$

Time limit

5 seconds.

Scoring

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