Southeastern European Regional Programming Contest
Bucharest, Romania
October 18, 2014

Problem A
Banks

Input File: A.in
Output File: standard output
Program Source File: A.c, A.cpp, A.java

On Wall Street from Wonderland, we have **n** banks, with $10000 > n > 0$. Each bank has exactly two neighbours, the left (*L*) and right (*R*) neighbour. The first bank's left neighbour is the last bank, while the last bank's right neighbour is the first bank. Each bank **i** $(n>i\geq 0)$ has a capital k_i with $32000>k_i>-32000$. The entire capital of all banks put together is known to be positive. Whenever some capital k_i of bank **i** is negative, the Bank Fairy can do a magic move and turn the capital into a positive one. For instance, if $k_i=-7$, after the magic move, $k_i=7$. Unfortunately, the magic move has consequences for both neighbours of bank **i**. Each sees its capital reduced with the absolute value of the capital of bank **i**. For instance if bank *L* has capital $k_L=5$ and bank *R* has capital $k_R=11$, then after the magic move $k_L=-2$ and $k_R=4$.

Which is the minimal number of magic moves which the Bank Fairy has to do in order to make the capital of all banks greater than or equal to 0?

On the first **input** line, we have the number **n** of banks. On the second input line, we have the capitals k_i $(n>i\geq 0)$ of all banks, in the order in which they are found on Wall Street from Wonderland. Each capital is separated by a single whitespace from the next one, except for the final capital which is directly followed by the newline character.

The **output** contains a single line with the value of the minimal number of magic moves.

Sample input	Sample output

`4`
`1 -2 -1 3`
	`9`

Problem B
Circle of digits

Input File: B.in
Output File: standard output
Program Source File: B.c, B.cpp, B.java

You are given a circular string of length **N** that consists of digits `'1'..'9'`. You want to split it into **K** continuous non-empty parts. Each of those parts represents a decimal notation of some integer number. Your goal is to find a partition that minimizes the maximum integer from the partition at hand.

For example, if the string is `7654321` and **K**=3 then the optimal partition is: `{176, 54, 32}` which has `176` as the maximum number. Note that the string is cyclic, that is the first character goes right after the last one (as in the `176` part of the above example).

Input
The first line of the input contains two integers **N** and **K** ($3 \le N \le 100000$, $2 \le K \le N$). The second line contains a string of length **N** which consists only of characters `'1'..'9'`.

Output
Output the value of the maximal number in the optimal partition.

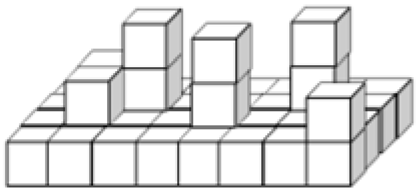
Sample input	Sample output
4 2 **4321**	**32**
7 3 **7654321**	**176**
5 5 **12321**	**3**

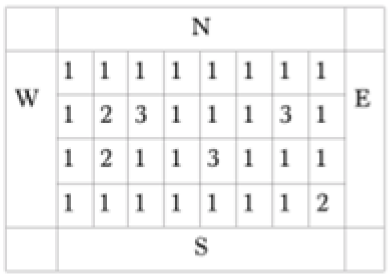
Comment
In the first sample the optimal partition is `{32, 14}`.
In the second sample the optimal partition is `{176, 54, 32}`.
In the third sample the optimal (and the only possible) partition is `{1, 2, 3, 2, 1}`.

Problem C
UFO

Input File: C.in
Output File: standard output
Program Source File: C.c, C.cpp, C.java

The security service of Snakeland wants to destroy the hostile alien ship. The security service has damaged the ship and forced it to land. The ship is built of cubic compartments of unit size. The first layer is always composed of **N * M** compartments. The picture (left) shows an example of a ship with size **N** = 4, **M** = 8, and (right) a top-view of the ship indicating the number of superimposed compartments.





	N	N	N	N	N	N	N	N	
W	1	1	1	1	1	1	1	1	E
	1	2	3	1	1	1	3	1	
	1	2	1	1	3	1	1	1	
	1	1	1	1	1	1	1	2	
	S	S	S	S	S	S	S	S	

Compartments are made of ultra-strong metal, so lasers are used to destroy the ship. Laser devices were deployed on each of the four sides of the ship, and they periodically produce beams perpendicular to the sides of the ship, towards different compartments of the ship. Each beam destroys the **R** first compartments on its way. If other compartments are located on top of destroyed ones, these compartments shift down.

After **K** shots it was decided to airstrike the ship. It makes sense to choose an area of size **P * P**, which contains a maximum number of remaining compartments to destroy them all.

Write a program that calculates maximum number of undivided compartments that can be destroyed by airstrike from an area with size **P * P**.

Input

The first line from the input contains 5 integers **N**, **M** ($1 \le N * M \le 1\ 000\ 000$), **R** ($0 < R \le 10$), **K** ($0 < K \le 300\ 000$), **P** ($0 < P \le \min(N, M, 10)$).

The following **N** lines contain **M** integers. The integer from the **i**-th row and **j**-column defines the number of compartments in the corresponding part of the ship as shown in the picture (right). Each integer is in the range 1..10^6.

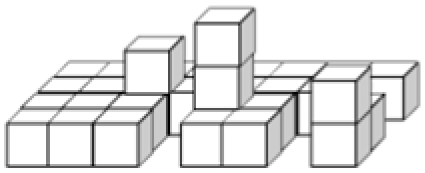
The next **K** lines describe laser shots. Each of these lines contains one symbol and 2 integers. Symbols define the side of the ship being shot: **W**, **E**, **S**, **N**. The first integer defines the row number in case of west or east and the column number in case of north or south, the second number indicates the horizontal layer being shot. Rows and columns are numbered as in the input, layers are numbered from `1`. Each integer is in the range `1..10`^{6}.

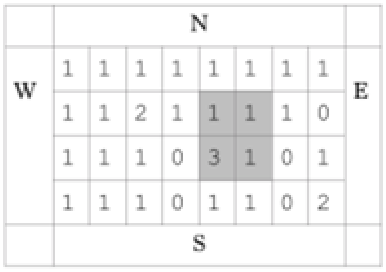
Output

You need to print the maximum number of remaining compartments after **`K`** laser shots contained in some area of size **`P * P`**.

Sample input	Sample output
`4 8 2 6 2`	**`6`**
`1 1 1 1 1 1 1 1`	
`1 2 3 1 1 1 3 1`	
`1 2 1 1 3 1 1 1`	
`1 1 1 1 1 1 1 2`	
`N 2 2`	
`W 2 2`	
`W 2 3`	
`E 2 1`	
`S 4 1`	
`S 7 1`	

Comment





				N					
W	1	1	1	1	1	1	1	1	E
	1	1	2	1	1	1	1	0	
	1	1	1	0	3	1	0	1	
	1	1	1	0	1	1	0	2	
				S					

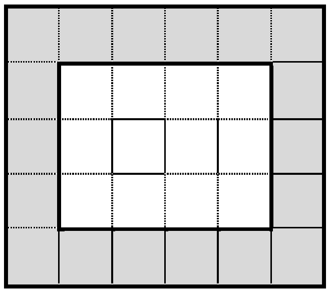
On the second picture you can see the state of the ship from the first picture after the shots described in sample input and one of `2*2` areas with the maximum number of remaining compartments.

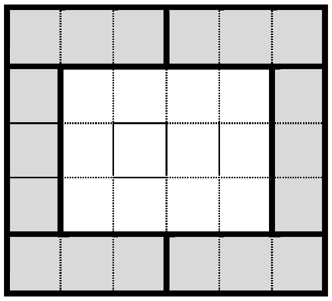
Southeastern European Regional Programming Contest
Bucharest, Romania
October 18, 2014

Problem D
Frame

Input File: D.in
Output File: standard output
Program Source File: D.c, D.cpp, D.java

Let's consider a $X \times Y$ rectangle with the middle $(X - 2) \times (Y - 2)$ rectangle cut out. We will call this figure a frame with size $X \times Y$. Picture 1 shows the frame 5×6.


Picture 1. Frame 5 × 6


Picture 2. Frame 5 × 6, paved with tiles 3 × 1

Let's assume that we have unlimited number of tiles with size $A \times 1$. We consider the following problem: is it possible to completely pave a frame with size $X \times Y$ using these tiles (tiles can be rotated by 90 degrees). For example, frame 5×6 from Picture 1 can be paved completely with tiles of size 3×1 (one way to do so is shown in Picture 2), but can't be paved with tiles of size 4×1.

Input
The first input line contains 2 integers – **X** and **Y** ($3 \le X \le 10^6$, $3 \le Y \le 10^6$). The second line contains integer **N** – the number of tile types to be analyzed ($1 \le N \le 1000$). Each of following **N** lines contains one integer, not exceeding 10^6. We designate with A_K the integer on the (k+2)-th line of the input file.

Output
Your goal is to print **N** lines, where the **K**-th line should contain the word "YES", if it is possible to tile the frame with size $X \times Y$ with tiles $A_K \times 1$, and the word "NO" otherwise.

Sample input	Sample output

5 6
2
3
4
	YES
	NO

Problem E
Points

Input File: E.in
Output File: standard output
Program Source File: E.c, E.cpp, E.java

Peter and Bob are playing a "Points" game on a math sheet of paper. Peter places a few points on the paper - grid nodes. Bob wants to surround them with a polygon so that all marked nodes are lying strictly within (not at the border) the polygon. All sides of the polygon are along the sides or the diagonals of the grid cells and its perimeter is as small as possible. You must determine what is the perimeter of the polygon.

Input
The first line of the input file contains integer **N** – the number of points placed by Peter ($1 \le N \le 100\,000$). Each of following **N** lines contains two integers x_i, y_i – the point coordinates placed by Peter. The coordinates by absolute value do not exceed 10^6. Some points can match.

Output
You need to print one number – the perimeter of the required polygon. The answer should be printed with accuracy not less than 0.001.

Sample input	Sample output
1 0 0	5.656

2
1 1
1 2
	7.656854

Problem F
Most Influential Pumpkin

Input File: F.in
Output File: standard output
Program Source File: F.c, F.cpp, F.java

Pumpkins in Hagrid's garden have come to life! Now they walk, talk, date … and, of course, organize elections to choose the Head Pumpkin! It turns out that it is pretty simple to guess who will win the next elections - it will be one of the pumpkins of the average size. To be precise, if all pumpkins line up in a row, sorted by their size, then the pumpkin exactly in the middle will be elected Head Pumpkin.

Hagrid does not want any fuss in his garden so he wants to know who is the Head Pumpkin. Of course, if there are several pumpkins of the same size, Hagrid doesn't know which one of them is the Head Pumpkin, but he is OK if he knows at least the size of the Head Pumpkin.

The pumpkins are growing in a row in the garden, conveniently numbered from 1 to **N**. Often, Hagrid waters some pumpkins that are growing together. More precisely, each time Hagrid selects two numbers **S** and **T** and waters all of the pumpkins between **S**-th and **T**-th in a row. After being watered, the pumpkins grow, increasing their size by exactly one. Of course, after watering, new elections happen and the size of the Head Pumpkin may change. Hagrid would like to know the size of the Head Pumpkin after each watering of the plants that he does.

Input
The input contains several test cases. The first line of each test contains two integers **N** and **K** ($1 \le N, K \le 60000$). **N** will always be an odd number. The sum of **N** for all tests will not exceed 60000. The sum of **K** for all tests will not exceed 60000.
The next line contains **N** integers A_i ($1 \le A_i \le 10^9$, $1 \le i \le N$) - the initial sizes of the pumpkins. The next **K** lines contain pairs of integers S_i and T_i ($1 \le S_i \le T_i \le N$, $1 \le i \le K$), indicating that Hagrid has watered all pumpkins with numbers between S_i and T_i. The last line of input contains two zeros. This line should not be processed or treated as a test case.

Output
For each test you should output **K** lines - the size of the Head Pumpkin after Hagrid has watered them first, second, ..., **K**-th time.

Sample input	Sample output

1 1
1
1 1
3 4
3 2 1
1 3
1 1
3 3
3 3
0 0
	2
	3
	3
	3
	4

`7 7`
`1 1 1 3 3 3 3`
`1 3`
`5 7`
`1 3`
`1 4`
`1 3`
`4 4`
`1 3`
`0 0`
	`3`
	`3`
	`3`
	`4`
	`4`
	`5`
	`5`

Problem G
Grammar

Input File: G.in
Output File: standard output
Program Source File: G.c, G.cpp, G.java

Bob is one of the best students of the Formal Languages class. Now he is learning about context free grammars in the Chomsky Normal Form (CNF). Such a grammar consists of:

- a set of nonterminal symbols *N*
- a set of terminal symbols *T*
- a special nonterminal symbol, called the start symbol *S*
- a set R of rules of the form $A \to BC$ or $A \to a$, where A, B, $C \in N$, $a \in T$.

If $A \in N$, we define $L(A)$, the language generated by A, as follows:

$$L(A) = \{ wz \mid w \in L(B), z \in L(C), \text{where } A \to BC \in R \} \cup \{ a \mid A \to a \in R \}.$$

The language generated by the grammar with start symbol *S* is defined to be $L(S)$.
Bob must solve the following problem: for a given context free grammar in CNF, on input string *x*, determine whether *x* is in the language generated by the grammar, L(S).

The program input is from a text file. It starts with the input string `x (|x|<=1000)`. Follows the grammar rules, in the form `ABC` or `Aa`, each on a separate line. We consider that the start symbol is always `S`. The program prints `1` if the string is in the language generated by the grammar, `0` otherwise.

The input data are correct and terminate with an end of file. The program prints the result to the standard output from the beginning of a line.

Input/output samples are given in the table below. There are three tests. The first two use the same grammar: `SAB`, `Sa`, `Ab` ($S \to AB$, $S \to a$, $A \to b$). For the first test the input string is `a`, and the result is `1`, while for the second test the input string is `ab` and the result is `0`.

Sample input	Sample output
a **SAB** **Sa** **Ab**	**1**
ab **SAB** **Sa** **Ab**	**0**
ab **SAB** **Sa** **Ab** **Ba**	**0**

Southeastern European Regional Programming Contest
Bucharest, Romania
October 18, 2014

Problem H
Triples

Input File: H.in
Output File: standard output
Program Source File: H.c, H.cpp, H.java

Mr. A invites you to solve the following problem:

"Let be $\boldsymbol{m}$ and $\boldsymbol{n}$ two positive integers, $\boldsymbol{5 \le m \le 100}$, $\boldsymbol{2 \le n \le 100}$. Consider the following sets of triples:

$$\boldsymbol{T_{m,j}} = \left\{(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \in \mathbb{N}^{\boldsymbol{3}} \;\middle|\; \boldsymbol{x \le y \le z \le m} \text{ and } \boldsymbol{x^j + y^j = z^j}\right\}, \qquad \boldsymbol{j = 2 \, .. \, n}$$

where $\mathbb{N}$ is the set of nonnegative integers ($\mathbb{N} = \{\boldsymbol{0}, \boldsymbol{1}, \boldsymbol{2}, \dots\}$).

The problem asks you to compute the sum $\boldsymbol{S_{m,n}}$:

$$\boldsymbol{S_{m,n}} = \sum_{\boldsymbol{j=2}}^{\boldsymbol{n}} \boldsymbol{card(T_{m,j})}$$

where $\boldsymbol{card(T_{m,j})}$ is the number of elements of the set $\boldsymbol{T_{m,j}}$."

Input
The input file contains a single test. The first line of the input file contains the value of m and the second line contains the value of n.

Output
The result will be written to standard output.

Sample input	Sample output

`85`
`95`
	`8128`

Southeastern European Regional Programming Contest
Bucharest, Romania
October 18, 2014

Problem I
Multi-Machine Scheduling of Two Applications

Input File: I.in
Output File: standard output
Program Source File: I.c, I.cpp, I.java

Little Ellie has two applications she needs to execute as quickly as possible. Application **i** (**i**=1,2) consists of **ns**(**i**) identical steps, numbered from 1 to **ns**(**i**). Ellie owns a cluster consisting of M machines (numbered from 1 to M) on which she can execute the steps of the 2 applications. The machines are not all identical, which means that the durations of executing one step on each machine may be different. To be more precise, it takes **T**(**i**,**j**) seconds to execute one step of application **i** on machine **j**. Because the steps of each application are very CPU-intensive, one machine can execute only one step of one application at one time (i.e. if multiple steps of the two applications are scheduled for execution on the same machine, the time intervals during which they are executed must be disjoint or only intersect at their endpoints). Moreover, the execution of any step of any application on any machine cannot be paused and then resumed later (on the same machine or on another machine). This means that once the execution of a step started on any machine, Ellie can either let the execution complete successfully or cancel it at any time before its completion and restart it later from the beginning (on the same machine or on any other machine).

Because of data dependencies, the steps of the same application must be executed sequentially (i.e. the execution of step **j** of application **i** can start only after the execution of step **j**-1 of that application has finished - either exactly at the same time when the execution of step **j**-1 finished or at any later time). The execution of step **j** can be scheduled either on the same machine on which step **j**-1 was executed or on any other machine. There are no dependencies between two steps of different applications.

Assuming that Ellie can start executing the steps of the two applications at time moment 0, she would like to minimize the time moment TEND (measured in seconds) when the last step of any application completes its execution. Please help her by telling her the minimum possible value of TEND.

Input Data

The first line of input contains the number of test cases T. The T test cases are described next. Each test case consists of 3 lines. The first line contains three integer numbers: **ns**(1), **ns**(2) and M. The second line contains M integers, representing, in order: **T**(1,1), **T**(1,2), ..., **T**(1,**M**). The third line contains **M** integers, representing, in order: **T**(2,1), **T**(2,2), ..., **T**(2,**M**).

Output Data

For each test case (in the order given in the input), output the minimum possible value of TEND on a separate line.

Constraints:

- $1 \le \mathbf{T} \le 20$
- $1 \le \mathbf{ns}(\mathbf{i}) \le 1000000$ (10^6)
- $1 \le \mathbf{M} \le 10$
- $1 \le \mathbf{T(i,j)} \le 1000$

Sample input	Sample output

`6`
`1000000 1000000 1`
`1`
`2`
`999999 999998 5`
`1 2 3 4 5`
`5 4 3 2 1`
`765432 765 2`
`1 2`
`1 1000`
`765432 766 2`
`1 2`
`1 1000`
`3 5 10`
`1 2 3 4 5 6 7 8 9 10`
`1 2 3 4 5 6 7 8 9 10`
`10 10 5`
`101 102 103 104 105`
`101 102 104 105 103`
	`3000000`
	`999999`
	`765432`
	`765433`
	`6`
	`1016`

Explanation

Example case 1: Since there is only one machine, all the steps of both applications are executed on the single available machine. The time needed for all of them to finish executing is `1000000*1+1000000*2=3000000.`

Example case 2: All the steps of application `1` are executed on machine `1` and all the steps of application `2` are executed on machine `5`. The time needed for all the steps to finish is `max{999999*1, 999998*1}=999999`.

Example case 3: All the steps of application `1` are executed on machine `1` and all the steps of application `2` are executed on machine `2`. The time needed for all the steps to finish is `max{765432*1, 765*1000}=765432.`

Example case 4: All the steps of application `1` are executed on machine `1` and the first `765` steps of application `2` are executed on machine `2`. After the first `765` steps of application `2` finish executing at the time moment `765000`, Ellie waits until time moment `765432`, when machine `1` becomes available. She then schedules the last step of application `2` on machine `1` and its execution finishes at the time moment `765433`.

Example case 5: All the steps of application `1` are executed on machine `2` and all the steps of application `2` are executed on machine `1`. The time needed for all the steps to finish is `max{3*2, 5*1}=6.`

Problem J
Strange Antennas

Input File: J.in
Output File: standard output
Program Source File: J.c, J.cpp, J.java

A President wants to survey its country. The country's territory is a grid of **n x n** cells. Every corner of each cell in the grid defines a point (indexed from **0** to **n** line and column-wise, `0< n <= 30000`). At every point of the grid the President can install an antenna.

Such an antenna emits a signal in one of the four directions: South-East (`SE`), South-West (`SW`), North-East (`NE`), North-West (`NW`). The shape of the signal is an isosceles right triangle. Each of the two equal sides of the triangle represents the antenna's power (`0< p <= 2*n`). The triangle defines the coverage of the antenna in the grid.

The President has already installed a number of antennas (we call this number **a**, `0 <= a <= 100`) and is interested in determining the total number of cells that are covered.

But these antennas are strange, because in any cell, if the signals coming from two antennas overlap, they cancel each other. In other words, for any given cell, the cell is going to be covered if and only if there is an odd number of signals overlapping in that cell.

For instance, given the `10` by `10` grid, with points ranging from `0` to `10`:

```
x x x x - - - - - -
x o o y y - - - - -
x x y y y - - - - -
x - - y y - - - - -
- - - - y - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
```

In the picture, the antenna `X` is situated on line `0`, column `0`, emitting a signal of power `4`, towards `South-East`. Antenna `Y` is situated on line `1`, column `5`, emitting of power `4`, towards `South-West`. In the above case, there is a total of `16` cells covered.

The **input** data has the following format: the number of cells **n** is followed by the number of antennas a, each on a separate line. The next **a** lines, for each antenna, contain: **line_position, column_position, power, orientation,** each on a separate line. The orientation is an integer in the range `0-3 (0:SW, 1:SE, 2:NE, 3:NW)`.

The **output** should be the number of cells covered by the antennas.

Sample input	Sample output

10
2
0 0 4 1
1 5 4 0
	16