NOI 2014 TASKS

Tasks
Task 1: ORCHARD
Task 2: SIGHTSEEING
Task 3: CATS
Task 4: OBELISK

Notes:

1. This document consists of 10 pages including this page.
2. Each task is worth 25 marks.
3. For each task, your program will be tested on a few sets of input instances. We call each set a subtask. Each subtask is worth a few marks. For each subtask, your program either obtains all the marks or none.
4. The subtasks vary in size and complexity, leading to different levels of difficulty. If you find solving the task completely difficult, you may want to focus on the easier subtasks.
5. The maximum execution time on each input instance in Task 1, 2, 3 & 4 is 1.0, 3.5, 1.5, 1.0 second respectively, and the memory size is limited to 256 MB.
6. Sample input and output files are provided. The file "`T`***`x`***`.0.in`" and "`T`***`x`***`.0.out`" are the corresponding input/output files described in the task statement. Under the system CMS, each sample input file is placed in the respective last subtask and worth 0 mark.

HAPPY PROGRAMMING!

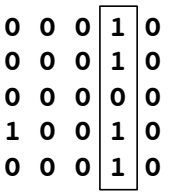
Task 1: ORCHARD

Alex and Bert are brothers who had been working for many years in a big orchard of their uncle where they planted trees. The orchard is arranged as an array of size n by m of trees. Alex had been planting apple trees and Bert had been planting banana trees; however, the brothers were not systematic and so apple trees stand among banana trees and vice versa. Each of them has planted at least one tree.

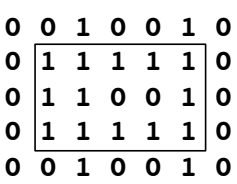
When coming near to retirement, the uncle decided to officially transfer the ownership of the trees to the brothers. The uncle informed the brothers that he will first pass the orchard to Alex. Next, Alex and Bert can cut out a rectangular area from the orchard, and the ownerships of all trees in the rectangular area are to be transferred to Bert. All further adjustments of the splitting had to be done with a lawyer.

Alex would like to keep all apple trees and Bert would like to keep all banana trees, but do not want to replant any tree. When they talked to a lawyer, the lawyer informed them that the ownership of a tree can be transferred from one owner to another, but the lawyer would charge \$1 to transfer the ownership of a tree. Therefore the brothers try to place the starting rectangle such that the legal fees are as low as possible.

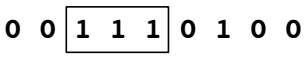
The following figures show three examples, where 0 and 1 indicates an apple and banana tree respectively.



Example 1



Example 2



Example 3

In the first example, the best way is to cut out the fourth column and to assign it to Bert, as indicated by the rectangular outline. Afterwards, there are two trees that are out of place, and their ownerships have to be transferred – one banana tree from Alex to Bert and one apple tree from Bert to Alex. Hence, the fees are \$2.

In the second example, the best way is to cut out trees that are not at the border of the field and then to transfer the ownership of six trees. The fees are \$6.

In the third example, the best way is to cut out the rectangle consisting of 3 banana trees, and then to transfer the ownership of the rightmost banana tree to Bert. The fee is \$1. Note that there is an alternative way that costs \$1 as well.

Input format

Your program must read from standard input. The first line of the input are the two positive numbers n and m indicating the orchard's size. Then follow n lines each consisting of m numbers, where 0 represents an apple tree and 1 represents a banana tree. For the second example, the input is as follow:

```
5 7
0 0 1 0 0 1 0
0 1 1 1 1 1 0
0 1 1 0 0 1 0
0 1 1 1 1 1 0
0 0 1 0 0 1 0
```

Output format

You program must write to the standard output a number, which is the smallest fee required. For the input above, the output is:

```
6
```

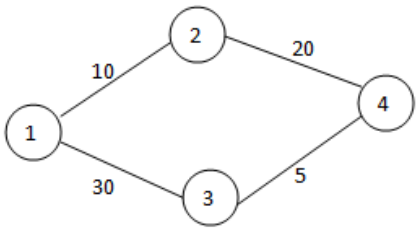
Subtasks

The maximum execution time on each instance is 1.0s. Your program will be tested on sets of input instances as follows:

1. (4 marks) Each instance satisfies $n = 1$ and $2 \leq m \leq 20$;
2. (4 marks) Each instance satisfies $n = 1$ and $2 \leq m \leq 15,000$;
3. (5 marks) Each instance satisfies $n = 1$ and $2 \leq m \leq 1,000,000$;
4. (3 marks) Each instance satisfies $1 \leq n \leq 2$ and $2 \leq m \leq 100,000$;
5. (4 marks) Each instance satisfies $1 \leq n \leq 150$ and $2 \leq m \leq 150$;
6. (5 marks) Each instance satisfies $1 \leq n \leq 150$ and $2 \leq m \leq 5,000$.

Task 2: `Sightseeing`

A travel agent wants to organize a sightseeing trip in a city for tourists. The city can be modelled as a connected graph, where each node represents a tourist site, and each edge represents a two-way road. Unfortunately, not all roads are good; some roads may be bad due to traffic jam. The agent does not want to disappoint the tourists by visiting the bad roads, and hence wants to compute the best path to take. He assigns a quality value to each road in such a way that a better road has a higher quality value. He also defines the quality value of a path to be the minimum quality among all roads along the path. The following figure depicts an example of 4 tourist sites and 4 roads modelled as a graph.



Each edge in the graph is associated with a value which represents the quality of the road. For e.g. the edge $(1, 2)$, which represents the road connecting node 1 and 2, has quality 10; while the edge $(3, 4)$ has quality 5. The quality of path 1-2-4 is the minimum among the two edges $(1, 2)$ and $(2, 4)$ and thus is $\min\{10, 20\} = 10$. Likewise, the quality of path 1-2-4-3 is the minimum among the three edges $(1, 2), (2, 4)$ and $(3, 4)$, which is $\min\{10, 20, 5\} = 5$.

Node 1 is the hotel where the tourists stay. Given a destination X, the travel agent wants to find the highest quality among all possible paths from node 1 to node X.

For instance, suppose they want to visit node 4. In the example above, among all possible paths from node 1 to node 4, the highest quality is achieved by the path 1-2-4, with quality 10. On the other hand, if they want to visit node 3 instead of node 4, the highest quality achievable is 30.

Furthermore, the travel agent is not satisfied in knowing the highest quality for a single destination. He has a list of destinations in mind, and he wants to know the highest quality for each of them. Specifically, when given a list of Q sites $X_1, X_2, \ldots, X_Q$, he wants to know the highest quality from node 1 to node X_1, the high quality from node 1 to node X_2, and so on.

Input format

Your program must read from the standard input. The first line in the input contains 3 integers, V, E and Q, which represent the number of tourist sites, the number of edges and the number of destinations respectively. The tourist sites are labelled from 1 to V where node 1 denotes the hotel where the tourists start their trip. Next, it is followed by E lines where each line contains 3 integers, v_1, v_2, and q, where v_1 and v_2 denote the sites that the road connects and q denotes the quality of the road ($0 \leq q \leq$ 100,000). Next, it is followed by Q lines where each line contains an integer X which represents a destination in the travel agent's list, and $X \neq 1$ (i.e. the hotel is not in the list). The input for the above example is:

```
4 4 2
1 2 10
1 3 30
2 4 20
3 4 5
3
4
```

Output format

Your program must write to standard output for each destination an integer which is the highest quality achievable. For the above input, the output is the following:

```
30
10
```

Subtasks

The maximum execution time on an input instance is 3.5s. Your program will be tested on 4 sets of input instances:

1. (4 marks) The roads form a connected simple cycle, hence, every node has exactly two edges. Furthermore, $V \leq 100$; $E \leq 100$; $Q \leq 50$.
2. (5 marks) The roads form a general connected graph. Furthermore, $V \leq 500$; $E \leq 4{,}000$; $Q \leq 100$.
3. (6 marks) Same as subtask 2, except that $V \leq 50{,}000$; $E \leq 100{,}000$; $Q \leq 10{,}000$.
4. (10 marks) Same as subtask 2, except that $V \leq 500{,}000$; $E \leq 5{,}000{,}000$; $Q \leq V - 1$.

Task 3: CATS

Esoteric languages are programming languages built around weird concepts either to test the limits of programming or for the purpose of comedy. One fine day, Mr. Panda was playing around with an esoteric language aptly called Counter and Two Stacks (CATS), whose name describes its basic usage. The language is designed to manipulate a counter ("COUNTER") and two stacks ("S1" and "S2") using several basic operations.

Under CATS, both stacks are initialised with an infinite number of 0's. Data can be pushed and popped from the stacks using the operations "PUSH" and "POP" respectively. An additional operation, "ADD", removes the top 2 elements of a stack, adds them and pushes the result back into the stack. Mr. Panda makes a novice attempt to draft some code that when given a query of 3 integers X, L and N, should print the X-th multiple of N larger than L. He uses negative numbers in his code and for convenience, Mr. Panda does not consider cases where L is a multiple of N.

Being new to programming, Mr. Panda's code does not print the correct answer. Furthermore, he did not know that the stacks in this language were not designed to work with negative integers. In CATS, whenever the code attempts to push a negative number using the PUSH operation, the number is not pushed into the stack. Instead CATS reacts unconventionally and all the numbers in that stack will have their last binary digit flipped (note that the stack contains an infinite number of values). For example, 4, whose binary representation is 100, would become 5, whose binary representation is 101, and vice versa. Flipping the last binary digit of a number X is equivalent to taking `(X xor 1)` in Pascal or `(X ^ 1)` in C/C++.

In addition, Mr. Panda's code mistakenly runs some operations twice. Eventually, Mr.Panda's code can be described by the pseudocode below. Here, "T1" and "T2" represent the top elements of S1 and S2 respectively.

```
COUNTER = X
WHILE COUNTER > 0
     S2 PUSH T1            //Push the top element of S1 onto S2
     S1 POP                //Pop the top element of S1
     FLIP LAST BINARY BIT OF ALL NUMBERS IN S1
     IF T2 > L
          COUNTER = COUNTER - 1
          IF COUNTER == 0 PRINT T2
     ELSE
          S2 PUSH N
          S2 PUSH N
          S2 ADD
          S2 ADD
          S1 PUSH T2
          S1 PUSH T2
          S2 POP
          S2 POP
```

Mr. Panda's code takes a long time to run and is thus difficult to debug. Can you help him figure out the output of his code quickly?

Input format

Your program must read from the standard input. Each input consists of multiple queries to Mr. Panda's code. The first line of each input contains a single positive integer Q which represents the number of queries. Each of the next Q lines contains 3 positive integers which represent the values of X, L and N respectively in a query. As stated above, L is not a multiple of N. An example is provided below:

```
2
4 5 2
18 6 4
```

Output format

For each query in the input, your program must write to the standard output the number printed by Mr. Panda's code. It is guaranteed that Mr. Panda's code prints exactly one integer for each query. Output the answer for different queries on different lines. For the above example, the output is:

```
8
9
```

Explanation

For the first query, the following lines trace the value of COUNTER and the values in S1 every time the WHILE loop is repeated. For simplicity, only the top 4 elements in S1 are shown. Here, the leftmost value on each line represents the top element of S1.

```
COUNTER: 4
S1: 0000...

COUNTER: 4
S1: 4411...

COUNTER: 4
S1: 8850...

COUNTER: 3
S1: 9411...

COUNTER: 2
S1: 5000...

COUNTER: 2
S1: 9911...

COUNTER: 1
S1: 8000...
```

In the next run, COUNTER will reach 0 so 8 is printed.

Subtasks

The maximum execution time on an input instance is 1.5s. Your program will be tested on 6 sets of input instances. Recall that all Q, X, L, and N are positive integers, and thus greater or equal to 1. The input instances satisfy the following conditions:

1. (4 marks)	$Q = 1$,	$X \leq 1,000,000$,	$L \leq 20$,	$N \leq 20$.
2. (4 marks)	$Q \leq 50$,	$X \leq 1,000,000$,	$L \leq 50$,	$N \leq 50$.
3. (4 marks)	$Q \leq 500$,	$X \leq 1,000,000$,	$L \leq 80$,	$N \leq 80$.
4. (4 marks)	$Q \leq 1,000$,	$X \leq 1,000,000$,	$L \leq 1,000,000$,	$N \leq 1,000,000$.
5. (4 marks)	$Q \leq 10,000$,	$X \leq 2^{30} - 1$	$L \leq 2^{30} - 1$,	$N \leq 2^{30} - 1$.
6. (5 marks)	$Q \leq 100,000$,	$X \leq 2^{60} - 1$	$L \leq 2^{60} - 1$,	$N \leq 2^{60} - 1$.

Task 4: OBELISK

A construction site consists of K floors, each of which can be treated as an infinitely large grid. On each floor, except the bottom floor, there are a number of holes, each one square large. At the bottom floor, there is a grid that is marked X. There is a heavy rectangular obelisk, of size $1 \times 1 \times M$, initially placed upright on the topmost floor. The construction workers would like to erect the obelisk vertically, such that one of the 1×1 faces is placed on the marked grid.

The obelisk is too heavy to be pushed or lifted, so the only way for the workers to move the obelisk is to roll it by tilting it along one of the edges as shown in Diagram 1.

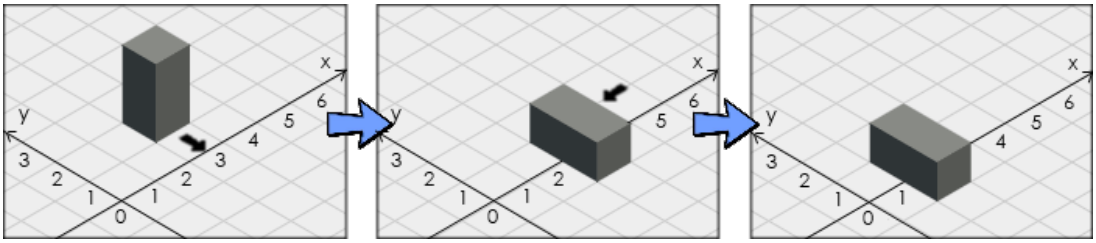


Diagram 1: Rolling an obelisk of size $1 \times 1 \times 2$

To move the obelisk from one floor to another floor below it, the workers will have to position the obelisk onto a hole, such that it falls to a lower floor, as shown in Diagram 2.

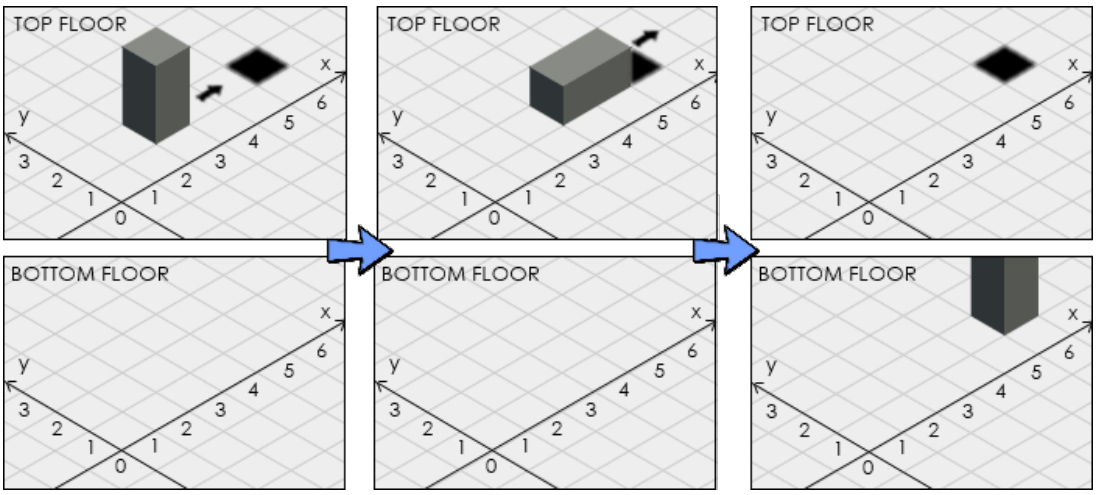


Diagram 2: Obelisk falling through a hole

As previously mentioned, each hole is 1×1 in size. No two holes are adjacent on the same floor. Hence, if the obelisk is horizontally placed over a hollow grid, it will not fall, unless M is 1. However, two or more holes in different layers may be vertically aligned – in this case the obelisk will fall through more than one layer, until it lands on a non-hole grid.

The obelisk is initially placed such that its edges are aligned with the grid in the construction area. Initially, the obelisk will always be placed vertically upright, with one of its 1×1 faces on a grid that is not a hole.

Your task is to find the the least number of moves to move the obelisk from its initial position on the top floor to the grid marked X on the bottom floor.

Input format

Your program must read from the standard input. The first line of the input contains 2 integers K and M, representing the number of floors and the height of the obelisk respectively.

The second line of the input contains four integers S_x, S_y, E_x, E_y, representing the starting x and y coordinates of the obelisk on the top floor and the ending x and y coordinates of the grid marked X on the bottom floor respectively.

Each of the next $K - 1$ lines describes the holes on each floor, starting from the top floor, down to the 2nd floor (i.e. the floor above the bottom floor). Each line begins with a positive integer h, indicating the number of holes on that floor, followed by a sequence of $2h$ space-separated integers $x_1, y_1, x_2, y_2, \ldots, x_h, y_h$. For each i, the integers x_i and y_i indicate the x and y coordinates of a hole on that floor. Note that every floor except the bottom floor has at least one hole.

An example is provided below:

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

Output format

Your program must write to the standard output a single integer, which is either the minimum number of moves needed to erect the given obelisk on the grid marked as X, or `-1` if it is not possible to move the obelisk onto the grid. Note that this answer may not fit in a 32-bit integer, requiring the use of long long in C/C++ or LongInt in Pascal. For the above example, the output is:

```
6
```

Subtasks

The maximum execution time on an input instance is 1.0s. Your program will be tested on 4 sets of input instances as follow:

1. (5 points) All instances in this set satisfy $M = 1$, $1 \leq K \leq 10$. All coordinates are positive integers no more than 50. There are at most 10 holes on each floor.
2. (7 points) All instances in this set satisfy $1 \leq M \leq 50$, $1 \leq K \leq 10$. All coordinates are positive integers no more than 50. There are at most 100 holes on each floor.
3. (6 points) All instances in this set satisfy $1 \leq M \leq 400$, $1 \leq K \leq 500$. All coordinates are positive integers no more than 400. There are at most 100 holes on each floor.
4. (7 points) All instances in this set satisfy $1 \leq M \leq 100000$, $1 \leq K \leq 500$. All coordinates are positive integers no more than $1,000,000,000$. There are at most 100 holes on each floor.