Arranging Tickets

In Republic of JOI, there are N stations numbered from 1 to N. They are located clockwise on a circular railway in order.

There are N types of train tickets numbered from 1 to N. By using one ticket of type i ($1 \leq i \leq N-1$), one person can travel from the station i to the station $i+1$, or from the station $i+1$ to the station i. By using one ticket of type N, one person can travel from the station 1 to the station N, or from the station N to the station 1. We can only buy a package of N tickets consisting of one ticket for each type.

You are working at a travel agency in Republic of JOI. Your task is to arrange tickets for customers.

Today, you have M requests for arranging tickets. The i-th request says C_i people want to travel from the station A_i to the station B_i. These C_i people need not to take the same route when they travel.

You want to know the minimum number of packages of tickets you need to buy in order to deal with all the requests.

Task

Given the number of stations and information of requests, write a program which calculates the minimum number of packages of tickets you need to buy.

Input

Read the following data from the standard input.

- The first line of input contains two space separated integers N, M. This means there are N stations in Republic of JOI, and you have M requests today.

- The i-th line ($1 \leq i \leq M$) of the following M lines contains three space separated integers A_i, B_i, C_i. This means the i-th request says C_i people want to travel from the station A_i to the station B_i.

Output

Write one line to the standard output. The output contains the minimum number of packages of tickets you need to buy.

Constraints

All input data satisfy the following conditions.

- $3 \leq N \leq 200\,000$.
- $1 \leq M \leq 100\,000$.
- $1 \leq A_i \leq N$ $(1 \leq i \leq M)$.
- $1 \leq B_i \leq N$ $(1 \leq i \leq M)$.
- $1 \leq C_i \leq 1\,000\,000\,000$ $(1 \leq i \leq M)$.
- $A_i \neq B_i$ $(1 \leq i \leq M)$.

Subtask

There are 5 subtasks. The score and additional constraints of each subtask are as follows:

Subtask 1 [10 points]

- $N \leq 20$.
- $M \leq 20$.
- $C_i = 1$ $(1 \leq i \leq M)$.

Subtask 2 [35 points]

- $N \leq 300$.
- $M \leq 300$.
- $C_i = 1$ $(1 \leq i \leq M)$.

Subtask 3 [20 points]

- $N \leq 3\,000$.
- $M \leq 3\,000$.
- $C_i = 1$ $(1 \leq i \leq M)$.

Subtask 4 [20 points]

- $C_i = 1$ $(1 \leq i \leq M)$.

Subtask 5 [15 points]

There are no additional constraints.

Sample Input and Output

Sample Input 1	Sample Output 1
3 3 1 2 1 2 3 1 3 1 1	1

If everybody travels clockwise, you need one ticket for each type. Hence you need to buy one package of tickets.

Sample Input 2	Sample Output 2
3 2 1 2 4 1 2 2	3

You need three tickets for each type if people travel in the following way:

- In the first request, three people travel clockwise, and one person travels counterclockwise.
- In the second request, two people travel counterclockwise.

Hence it is enough to buy three packages of tickets.

We output 3 because it is impossible to travel if you buy only two packages.

Sample Input 3	Sample Output 3
6 3 1 4 1 2 5 1 3 6 1	2

For example, you buy two packages of tickets, and distribute them in the following way:

- Give tickets $1, 2, 3$ to the person who wants to travel from the station 1 to the station 4.
- Give tickets $1, 6, 5$ to the person who wants to travel from the station 2 to the station 5.
- Give tickets $3, 4, 5$ to the person who wants to travel from the station 3 to the station 6.

We output 2 because it is impossible to travel if you buy one package only.

Broken Device

Anna and Bruno are archaeologists. They are investigating ruins in Iran.

Their tasks are as follows: Anna visits ruins and discovers artifacts, and Bruno analyzes the results in the base camp.

Their investigation is scheduled for $Q(= 1\,000)$ days. Every day, Anna sends the results to Bruno using a communication device. The results for each day is expressed as an integer X.

Anna can use the communication device once per day only. It can send a sequence of length $N(= 150)$ consisting of 0 or 1.

However, it was broken. There are several broken places in the sequence of length N. For a broken place, it always sends the value 0 regardless of the actual set value. When Anna sends a sequence, she can see the positions of the broken places. But, Bruno does not know them. The positions of broken places and the number of broken places can change every day.

There is a danger that their investigation will be delayed. Because you are a candidate of a contestant of an international programming contest in Iran, Anna and Bruno ask you to write a program which sends the results of their investigation.

Task

Write two programs which achieve the communication between Anna and Bruno.

- Given the length of the sequence N, the integer to be sent X, the number of broken places K, and the positions of broken places P, the first program sets the sequence S sent by Anna.
- Given the sequence A received by Bruno, the second program recovers the integer X.

At a place where the communication device works, the sequence S and the sequence A have the same value. At a broken place, the sequence A always has the value 0 regardless of the values of the sequence S.

Implementation Details

You need to submit two files written by the *same programming language*.

The first file is either `Anna.c` or `Anna.cpp`. This file sets a sequence sent by Anna, and implements the following function. The program should include `Annalib.h`.

- `void Anna( int N, long long X, int K, int P[] )`

  For each test case, this function is called $Q = 1\,000$ times.

  - The parameter `N` is the length of the sequence to be sent.

- The parameter `X` is the integer to be sent.
- The parameter `K` is the number of broken places.
- The parameter `P[]` is a sequence of length K, describing the positions of broken places.

In the function `Anna`, the following function must be called.

★ `void Set( int pos, int bit )`

This function sets a bit in the sequence S to be sent by the communication device.

- The parameter `pos` is the position of the place to be set. `pos` must be an integer between 0 and $N - 1$, inclusive. *Note that the positions are counted from* 0. If it is called with parameter outside this range, your program is considered as **Wrong Answer[1]**. It is not allowed to call this function more than once with the same parameter `pos`. If it is called more than once with the same parameter, your program is considered as **Wrong Answer[2]**.
- The parameter `bit` is the value set to the `pos`-th position of the sequence. The value of `bit` must be either 0 or 1. If it is called with other parameters, your program is considered as **Wrong Answer[3]**.

The function `Set` must be called exactly N times in the function `Anna`. When function `Anna` terminates, if the number of times the function `Set` is called is different from N, your program is considered as **Wrong Answer[4]**.

Your program is terminated if the function call by `Anna` is considered invalid.

The second file is either `Bruno.c` or `Bruno.cpp`. This file recovers the integer expressing the results of investigation, and implements the following function. The program should include `Brunolib.h`.

- `long long Bruno( int N, int A[] )`

  For each test case, this function is called $Q = 1\,000$ times.

  - The parameter `N` is the length of the sequence received by Bruno.
  - The parameter `A` is an integer sequence of length N. It is the sequence received by Bruno.
  - The function `Bruno` must recover the value of X, and return it.

Grading Procedure

The grading is done in the following way. If your program is considered as Wrong Answer, it is terminated immediately.

(1) Set `cnt` = 0.

(2) Call the function `Anna` once.

(3) Let S be the sequence set by the function `Anna`. In the sequence S, set the value 0 to the positions in P, and obtain the sequence A. Call the function `Bruno` once with parameter A.

(4) Set `cnt` = `cnt` +1. If `cnt` < Q, go to (2). If `cnt` = Q, go to (5).

(5) Your program will be scored.

Important Notices

- Running time and memory usage are calculated for (1), (2), (3), (4) of Grading Procedure.
- Your program must not be considered as Wrong Answer in the function call to `Anna` in (2) or `Bruno` in (3). Your program must be executed without runtime error.
- Your program can implement other functions for internal use, or use global variables. Submitted programs will be compiled with the grader, and become a single executable file. All global variables and internal functions should be declared `static` to avoid confliction with other files. Since the programs of Anna and Bruno will be executed as 2 distinct processes when they are graded, they can not share global variables.
- In the process, each of the functions `Anna` and `Bruno` is called $Q = 1\,000$ times. *The variables used by your programs should be initialized appropriately.*
- Your program should not use standard input and standard output. Your program should not communicate with other files by any methods.

Compilation and Test Run

You can download an archive file from the contest webpage which contains a sample grader to test your program. The archive file also contains a sample source file of your program.

A sample grader consists of one source file which is either `grader.c` or `grader.cpp`. For example, if your programs are `Anna.c` and `Bruno.c`, or `Anna.cpp` and `Bruno.cpp`, you run the following commands to compile your programs.

- C

```
gcc -std=c11 -O2 -o grader grader.c Anna.c Bruno.c -lm
```

- C++

```
g++ -std=c++14 -O2 -o grader grader.cpp Anna.cpp Bruno.cpp
```

When the compilation succeeds, the executable file `grader` is generated.

Note that the actual grader is different from the sample grader. The sample grader will be executed as a single process, which will read input data from the standard input and write the results to the standard output.

Input for the sample grader

The sample grader reads the following data from the standard input.

- The first line contains an integer Q.
- Then, information of Q queries are given.
- Information of each query consists of two lines as follows.
  - The first line contains three space separated integers N, X, K. This means the length of the sequence to be sent is N, the integer to be sent by Anna is X, and there are K broken places.
  - The second line contains K space separated integers $P_0, P_1, \cdots, P_{K-1}$. This means, for each i ($0 \leq i \leq K-1$), the P_i-th place in the sequence is broken.

Output of the sample grader

When the program terminates successfully, the sample grader writes the following information to the standard output. (The quotation mark is not written actually.)

- If your program is considered as Wrong Answer, the sample grader writes its type in the following form "Wrong Answer [1]," and your program is terminated.
- If every function call to `Anna` is not considered as Wrong Answer, the sample grader writes "`Accepted`" and the value of L^*. For the value of L^*, see Scoring.

If your program is considered as several types of Wrong Answer, the sample grader reports only one of them.

Constraints

All input data satisfy the following conditions.

- $Q = 1000$.
- $N = 150$.
- $0 \leq X \leq 1\,000\,000\,000\,000\,000\,000$.
- $1 \leq K \leq 40$.
- $0 \leq P_i \leq N-1$ ($0 \leq i \leq K-1$).
- $P_i < P_{i+1}$ ($0 \leq i \leq K-2$).

Scoring

- Let L^* be the minimum of the following value for all test cases of this task.
  - The maximum integer $L \leq 40$ such that, for every query with $K \leq L$, Bruno answers the correct value of X.
- The score of this task is calculated by the following way:
  - If $L^* = 0$, the score is 0 point.
  - If $1 \leq L^* \leq 14$, the score is 8 points.
  - If $15 \leq L^* \leq 37$, the score is $(L^* - 15) \times 2 + 41$ points.
  - If $38 \leq L^* \leq 40$, the score is $(L^* - 38) \times 5 + 90$ points.

Sample Communication

Here is a sample input for grader and corresponding function calls. Note that the following example does not satisfy the constraints of this task because $Q = 2, N = 3$.

Sample Input	Sample Calls			
	Call	Return	Call	Return
2 3 14 1 2 3 9 2 0 1	`Anna(...)`			
			`Set(0,0)`	
				(none)
			`Set(1,0)`	
				(none)
			`Set(2,1)`	
				(none)
		(none)		
	`Bruno(...)`			
		14		
	`Anna(...)`			
			`Set(0,0)`	
				(none)
			`Set(1,1)`	
				(none)
			`Set(2,1)`	
				(none)
	`Bruno(...)`			
		9		

Here the parameters for `Anna(...)`, `Bruno(...)`, `Anna(...)`, `Bruno(...)` are as follows.

Parameter	`Anna(...)`	`Bruno(...)`	`Anna(...)`	`Bruno(...)`
`N`	3	3	3	3
`X`	14		9	
`K`	1		2	
`P`	{2}		{0, 1}	
`A`		{0, 0, 0}		{0, 0, 1}

Railway Trip

JOI Railways is a company operating one railway. In the railway of JOI Railways, there are N stations on a straight line numbered from 1 to N. For each i ($1 \leq i \leq N-1$), the station i and the station $i+1$ are connected by a railway track.

JOI Railways has K types of trains running in both directions. The types of trains are expressed as integers between 1 and K inclusive. Each station has a *level* which is an integer between 1 and K inclusive. For each i ($1 \leq i \leq N$), the station i has level L_i. The stations in both ends, namely the station 1 and the station N, have level K.

A train of type j ($1 \leq j \leq K$) stops at every station whose level is greater than or equal to j, and it does not stop at any other stations. Since the stations in both ends, namely the station 1 and the station N, have level K, every train stops at these stations.

Many passengers use JOI Railways every day. During travel, they can take a train whose direction is opposite to the destination, or they can pass the destination. In the end of the travel, they have to stop at the destination. They do not like to stop at stations very much. Hence they try to take a route with minimum number of intermediate stops regardless of the number of passed stations or the number of connections. If a passenger stops at a station to change trains, we count it as one stop. The first stop at the starting station and the last stop at the destination are not considered as intermediate stops.

Your task is to write a program which answers queries on the minimum number of intermediate stops for each passenger.

Task

Given information of the railway of JOI Railways and the starting station and the destination of each passenger, write a program which answers queries on the minimum number of intermediate stops for each passenger.

Input

Read the following data from the standard input.

- The first line of input contains three space separated integers N, K, Q. This means there are N stations in JOI Railways, there are K types of trains, and Q queries about travels between two stations are given.
- The i-th line ($1 \leq i \leq N$) of the following N lines contains an integer L_i, the level of the station i.
- The k-th line ($1 \leq k \leq Q$) of the following Q lines contains two space separated integers A_i, B_i. This means the starting station and the destination of the k-th passenger are the stations A_i, B_i, respectively.

Output

Write Q lines to the standard output. The k-th line ($1 \leq k \leq Q$) of output contains the minimum number of intermediate stops of a route from the station A_k to the station B_k.

Constraints

All input data satisfy the following conditions.

- $2 \leq N \leq 100\,000$.
- $1 \leq K \leq N$.
- $1 \leq Q \leq 100\,000$.
- $1 \leq L_i \leq K$ ($1 \leq i \leq N$).
- $1 \leq A_k \leq N$ ($1 \leq k \leq Q$).
- $1 \leq B_k \leq N$ ($1 \leq k \leq Q$).
- $A_k \neq B_k$ ($1 \leq k \leq Q$).

Subtask

There are 4 subtasks. The score and additional constraints of each subtask are as follows:

Subtask 1 [5 points]

- $N \leq 100$.
- $K \leq 100$.
- $Q \leq 50$.

Subtask 2 [15 points]

- $Q \leq 50$.

Subtask 3 [25 points]

- $K \leq 20$.

Subtask 4 [55 points]

There are no additional constraints.

Sample Input and Output

Sample Input 1	Sample Output 1
9 3 3 3 1 1 1 2 2 2 3 3 2 4 4 9 6 7	1 3 0

In this sample input, three queries about travels between two stations are given.

- The first query is about the travel from the station 2 to the station 4. If a passenger takes a train of type 1 from the station 2 to the station 4, there is only one intermediate stop, the station 3.
- The second query is about the travel from the station 4 to the station 9. If a passenger takes a train of type 1 from the station 4 to the station 5, then a train of type 2 from the station 5 to the station 1, and finally a train of type 3 from the station 1 to the station 9, there are three intermediate stops, the stations 5, 1, 8.
- The third query is about the travel from the station 6 to the station 7. If a passenger takes a train of type 2 from the station 6 to the station 7, there are no intermediate stops.

Sample Input 2	Sample Output 2
5 2 1 2 1 1 1 2 1 4	1

Note that passengers can pass the destination during travel.

Sample Input 3	Sample Output 3
15 5 15 5 4 1 2 3 1 1 2 4 5 4 1 5 3 5 8 1 11 1 5 3 6 11 9 12 15 14 15 2 3 12 2 1 4 8 15 5 12 6 1 13 13 8 14 9	2 1 1 3 2 0 3 4 0 1 3 4 1 2 2