문제 1. 티켓 정리

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	4초
메모리 제한:	256MB

JOI 공화국에는 1번부터 N번까지 번호가 붙은 N 개의 기차역이 있다. 기차역은 원형 철도에 시계방향으로 위치해 있다.

철도를 이용하기 위한 티켓은 N 종류가 있고 1번부터 N번까지 번호가 붙어 있다. i번 ($1 \leq i \leq N-1$) 티켓을 하나 사용하면 i번 기차역에서 $i+1$번 기차역으로 혹은 $i+1$번 기차역에서 i번 기차역으로 사람 한 명이 이동할 수 있다. N번 티켓을 하나 사용하면 1번 기차역에서 N번 기차역으로 혹은 N번 기차역에서 1번 기차역으로 사람 한 명이 이동할 수 있다. 이 티켓은 N 종류의 티켓이 정확히 한 장씩 총 N 장이 들어 있는 묶음으로 판매되고 있다.

당신은 JOI 공화국의 여행회사에서 일하고 있다. 당신은 고객들에게 티켓을 나눠주어야 한다.

오늘 티켓을 나눠달라는 M 개의 요청이 있었다. i 번째 요청은 C_i 명의 사람이 A_i번 기차역에서 B_i번 기차역으로 이동하고 싶다는 요청이었다. C_i 명의 사람들이 모두 같은 경로로 이동 할 필요는 없다.

모든 요청을 처리하기 위해서 사야 할 묶음의 최소 개수를 알고 싶다.

입력 형식

다음 정보가 표준 입력으로 주어진다.

- 첫째 줄에는 공백으로 구분된 두 정수 N, M이 주어진다. JOI 공화국에 N 개의 기차역이 있으며 M 개의 요청을 오늘 받았다는 의미이다.
- 다음 N 개의 줄의 i 번째 ($1 \leq i \leq N$) 줄에는 공백으로 구분된 세 정수 A_i, B_i, C_i가 주어진다. 이는 i 번째 요청이 C_i명의 사람이 A_i번 기차역에서 B_i번 기차역으로 이동하고 싶다는 요청이라는 의미이다.

출력 형식

표준 출력으로 한 개의 줄을 출력하여라. 출력은 모든 요청을 처리하기 위해서 사야 할 묶음의 최소 개수이다.

제한

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

서브태스크 1 (10 점)

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

서브태스크 2 (35 점)

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

서브태스크 3 (20 점)

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

서브태스크 4 (20 점)

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

서브태스크 5 (15 점)

추가 제한조건이 없다.

예제

standard input	standard output
3 3 1 2 1 2 3 1 3 1 1	1

모두가 시계방향으로 이동하면 각 종류의 티켓이 하나씩 필요하다. 즉, 한 묶음만 사도 충분하다.

standard input	standard output
3 2 1 2 4 1 2 2	3

다음 방법으로 이동하면 각 종류의 티켓이 세 장씩 필요하다:

- 첫 번째 요청에서, 세 명이 시계방향으로, 한 명이 반시계방향으로 움직인다.
- 두 번째 요청에서, 두 명이 반시계방향으로 움직인다.

그래서 세 묶음을 사면 충분하다.
두 묶음을 사서 이동하는 것은 불가능하므로 3을 출력한다.

standard input	standard output
6 3 1 4 1 2 5 1 3 6 1	2

예를 들면 두 묶음을 사서 다음과 같이 나누어 주면 된다.

- 1번 역에서 4번 역으로 이동하고 싶은 사람에게 1, 2, 3번 티켓을 준다.
- 2번 역에서 5번 역으로 이동하고 싶은 사람에게 1, 6, 5번 티켓을 준다.
- 3번 역에서 6번 역으로 이동하고 싶은 사람에게 3, 4, 5번 티켓을 준다.

한 묶음을 사서 이동하는 것은 불가능하므로 2를 출력한다.

문제 2. 고장난 기기

시간 제한: 2초
메모리 제한: 256MB

고고학자 Anna와 Bruno는 이란의 유적을 조사하고 있다.

두 명이 역할을 나눠 Anna는 유적을 발견하고 유물을 발견하며 Bruno는 베이스캠프에서 결과를 분석한다.

조사는 총 $Q(= 1\ 000)$일 동안 계획이 되어있다. 매일 Anna는 Bruno에게 통신 기기를 사용하여 결과를 보낸다. 각 통신기기의 결과는 정수 X로 표시된다.

안나는 통신 기기를 하루에 한 번만 사용할 수 있다. 통신기기는 0 혹은 1로 되어있는 길이 $N(= 150)$의 수열을 보낼 수 있다.

하지만 이 기계가 고장 나 버려서 보내는 길이 N의 수열 중 기능이 제대로 작동하지 않는 위치가 생겨버렸다. 기능하지 않는 위치에는 어떤 값을 설정해도 0이 보내지게 된다. Anna가 수열을 보낼 때 고장 난 위치가 어딘지 알 수 있다. 하지만, Bruno는 그 위치를 모른다. 고장 난 위치가 몇 개인지, 어딘지는 매일 바뀐다.

이 조사가 지연되면 문제가 생길 수 있다. Anna와 Bruno는 이란에서 열리는 국제 프로그래밍 대회의 후보인 당신에게 조사 결과를 보내는 프로그램을 작성해달라고 요청했다.

당신은 Anna와 Bruno 사이에서 통신하는 두 개의 프로그램을 작성하여야 한다.

- 수열의 길이 N, 보내야 할 정수 X, 고장 난 위치의 개수 K, 고장 난 위치 P가 주어졌을 때, 첫 번째 프로그램은 Anna가 보낼 수열 S를 정한다.
- Bruno가 수열 A를 받았을 때, 두 번째 프로그램은 정수 X를 복구한다.

통신 장치가 고장 난 위치가 아닌 곳에서는, 수열 S와 수열 A는 같은 값을 가진다. 통신 장치가 고장 난 곳에서는, 수열 A는 수열 S의 값과 관계없이 0이다.

구현 명세

당신은 *같은 프로그래밍 언어로 작성된* 파일 두 개를 작성해야 한다.

첫 번째 파일의 이름은 `Anna.c` 혹은 `Anna.cpp`이다. 이 파일은 Anna가 보낼 수열을 정하는 역할을 하며, 다음 함수를 구현해야 한다. 이 파일은 `Annalib.h`를 include해야 한다.

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

  이 함수는 각 테스트 케이스마다 정확히 $Q = 1\ 000$ 번 불린다.

  - 인자 `N`은 보낼 수열의 길이를 나타낸다.
  - 인자 `X`는 보낼 숫자를 나타낸다.
  - 인자 `K`는 부서진 위치의 개수를 나타낸다.
  - 인자 `P[]`는 부서진 위치를 나타내는 길이 K의 수열이다.

  함수 `Anna`는 다음 함수를 호출해야 한다.

  - `void Set(int pos, int bit)`
    이 함수는, 통신 기기로 보낼 수열 S를 설정한다.
    - 인자 `pos`는 수열의 값을 설정할 위치이다. `pos`는 0 이상 $N - 1$ 이하이다. *위치가 0부터 시작함에 유의하여라.* 만약에 이 범위를 벗어나서 함수를 호출한 경우 **오답** [1]이 된다. 같은 `pos`를 인자로 하여 함수를 두 번 이상 호출한 경우 **오답** [2]이 된다.
    - 인자 `bit`는 `pos` 번째에 설정할 값이다. `bit`의 값은 0 혹은 1이어야 한다. 다른 인자로 함수를 호출한 경우 **오답** [3]이 된다.

  함수 `Set`은 함수 `Anna` 안에서 정확히 N번 호출되어야 한다. `Anna` 함수가 종료되었을 때, `Set`이 호출된 횟수가 N과 다르면 **오답** [4]이 된다.

  만약 `Anna`가 함수를 올바르지 않게 호출할 경우 프로그램이 종료된다.

두 번째 파일의 이름은 `Bruno.c` 혹은 `Bruno.cpp`이다. 이 파일은 탐사 결과를 복구하는 역할을 하며, 다음 함수를 구현해야 한다. 이 파일은 `Brunolib.h`를 include해야 한다.

- `long long Bruno(int N, int A[])`
  이 함수는 각 테스트 케이스마다 정확히 $Q = 1\ 000$ 번 불린다.
  - 인자 `N`은 Bruno가 받은 수열의 길이를 나타낸다.
  - 인자 `A[]`는 Bruno가 받은 길이 N의 수열이다.
  - 함수 Bruno는 X를 찾아서 반환해야 한다.

채점은 다음과 같은 방식으로 진행된다. 만약 프로그램이 오답으로 판단된 경우, 즉시 채점은 종료된다.

1. `cnt=0`으로 설정한다.
2. 함수 `Anna`를 한 번 호출한다.
3. 함수 `Anna`에 호출된 수열을 S라고 하자. S중 P에 포함된 위치를 0으로 바꾸는 작업을 `A`에 한 후, 함수 `Bruno`를 한 번 호출한다.
4. `cnt=cnt+1`로 설정한다. `cnt`<Q이면 2.로 돌아간다. `cnt`=Q이면, 5.로 간다.
5. 채점이 완료된다.

참고 사항

- 실행 시간과 메모리 사용량은 채점 방식의 1, 2, 3, 4에서 계산된다.
- 당신의 프로그램은 채점 방식 2.의 `Anna` 혹은 채점 방식 3.의 `Bruno`에서 오답으로 판단되면 안 된다. 당신의 프로그램은 런타임 에러 없이 실행되어야 한다.
- 당신의 프로그램은 내부에서 사용할 목적으로 함수나 전역변수를 사용할 수 있다. 제출한 프로그램은 그레이더와 함께 컴파일되어 하나의 실행파일이 된다. 모든 전역변수나 내부 함수는 다른 파일과의 충돌을 피하기 위해 `static`으로 선언되어야 한다. Anna와 Bruno는 2개의 별개의 프로세스로 실행되기 때문에 채점될 때 전역변수를 공유하지 않는다.
- 각 프로세스에서, `Anna`와 `Bruno`는 각각 $Q = 1\ 000$번 호출된다. **사용할 변수의 초기화는 적절히 진행되어야 한다.**
- 당신의 프로그램은 표준 입출력을 사용해서는 안 된다. 당신의 프로그램은 어떠한 방법으로도 다른 파일에 접근해서는 안 된다.

당신은 대회 홈페이지의 아카이브에서 프로그램을 테스트하기 위한 목적의 샘플 그레이더를 받을 수 있다. 아카이브는 당신의 프로그램의 예제 소스 또한 첨부되어 있다. 샘플 그레이더는 파일 `grader.c` 혹은 `grader.cpp`이다. 당신의 프로그램이 `Anna.c`와 `Bruno.c` 혹은, `Anna.cpp`와 `Bruno.cpp` 인 경우 다음 커맨드로 컴파일할 수 있다.

- C `g++ -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`

컴파일이 성공적이면, 파일 `grader`가 생성된다.

실제 그레이더와 샘플 그레이더는 다름에 주의하여라. 샘플 그레이더는 하나의 프로세스에서 실행되며, 입력을 표준 입력으로부터 받고, 출력을 표준 출력에 출력한다.

입력 형식

샘플 그레이더는 다음 형식으로 표준 입력으로 부터 데이터를 입력받는다.

- 첫째 줄에 정수 Q가 주어진다.
- 그리고 Q개의 쿼리의 정보가 주어진다.
- 각 쿼리의 정보는 다음과 같은 두 줄로 되어있다.
  - 첫 번째 줄은 공백으로 구분된 세 정수 N, X, K가 주어진다. 이는 보낼 수열의 길이가 N이고, Anna가 보낼 정수가 X고, 부서진 위치가 K개라는 의미이다.
  - 두 번째 줄은 공백으로 구분된 K개의 정수 $P_0, P_1, \cdots, P_{K-1}$ 이 주어진다. 이는 각 i $(0 \leq i \leq K-1)$에 대해, 수열의 P_i번째 위치가 고장 났다는 의미이다.

출력 형식

프로그램이 정상적으로 종료되었다면, 샘플 그레이더는 다음과 같은 정보를 표준 출력에 출력한다. (따옴표는 출력하지 않는다.)

- 오답으로 판단된 경우, 오답의 종류를 "`Wrong Answer [1]`"과 같은 형식으로 출력하고, 프로그램이 종료된다.
- 만약 모든 Anna의 호출이 오답으로 판단되지 않을 경우, "`Accepted`"와 L^*을 표준 출력으로 출력한다. L^*의 값은 배점 항목을 참고하여라.

프로그램이 다양한 오답의 종류에 속해 있으면 샘플 그레이더는 그중 하나만 출력할 것이다.

제한

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

배점

- L^*를 이 문제의 모든 테스트 케이스의 최솟값이라고 하자.
  - $K \leq L$인 모든 쿼리에 대해서 Bruno가 정답을 말한 최대 정수 $L \leq 40$.
- 이 문제의 점수는 다음과 같이 계산된다.
  - $L^* = 0$인 경우, 점수는 0점이다.
  - $1 \leq L^* \leq 14$인 경우, 점수는 8점이다.
  - $15 \leq L^* \leq 37$인 경우, 점수는 $(L^* - 15) \times 2 + 41$점 이다.
  - $38 \leq L^* \leq 40$인 경우, 점수는 $(L^* - 38) \times 5 + 90$점 이다.

예제

예제 입력과 이에 해당하는 함수 호출을 보여준다. 이 예제는 $Q = 2, N = 3$이기 때문에 문제의 제한을 만족하지 않음에 유의하여라.

예제 입력	예제 함수 호출			
	호출	반환값	호출	반환값
2 3 14 1 2 3 9 2 0 1	`Anna(...)`			
			`Set(0,0)`	
				(없음)
			`Set(1,0)`	
				(없음)
			`Set(2,1)`	
				(없음)
		(없음)		
	`Bruno(...)`			
		`14`		
	`Anna(...)`			
			`Set(0,0)`	
				(없음)
			`Set(1,1)`	
				(없음)
			`Set(2,1)`	
				(없음)
		(없음)		
	`Bruno(...)`			
		`9`		

여기서 `Anna(...)`, `Bruno(...)`, `Anna(...)`, `Bruno(...)` 호출의 인자들은 다음과 같다.

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

문제 3. 철도 여행

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	2초
메모리 제한:	512MB

JOI 전철은 철도 하나를 운영하는 회사이다. JOI 전철의 철도에는 1번부터 N번까지의 번호가 붙어있는 N 개의 철도역이 일직선상에 놓여있다. 각 i $(1 \leq i \leq N-1)$에 대해서, i번 역과 $i+1$번 역은 선로로 연결되어있다.

JOI 전철에는 양방향으로 달리는 K 종류의 열차가 있다. 열차의 종류는 1 이상 K 이하의 정수로 표현된다. 각 역에는 **중요도**라고 하는 1 이상 K 이하의 정수가 하나씩 붙어 있다. i번 $(1 \leq i \leq N)$ 역의 중요도는 L_i 이다. 양 끝에 있는, 즉 1번과 N번 역은 중요도가 K 이다.

j번 $(1 \leq j \leq K)$ 종류의 열차는 중요도가 j 이상인 철도역에서만 멈추고 다른 철도역에서는 멈추지 않는다. 양 끝에 있는 1번과 N번 철도역은 중요도가 K이기 때문에 모든 열차는 이 철도역에서는 멈춘다.

많은 승객이 JOI 전철을 이용하고 있다. 여행 중에 승객들은 목적지와 반대 방향으로 움직이거나 목적지를 통과 할 수도 있지만 결국에는 목적지에 멈추어야 한다. 승객들은 역에 멈추는 것을 좋아하지 않는다. 그러므로 도중에 정차하는 철도역의 수를 최소화하고 싶다. 어떤 승객이 열차를 갈아타기 위해 역에 멈춘 경우에도 한 번으로 계산하고, 출발역과 도착역은 세지 않는다.

당신의 업무는 승객의 출발역과 도착역이 주어졌을 때, 이 두 역의 사이를 이동하는 도중에 정차하는 철도역의 수를 구하는 프로그램을 작성해야 한다.

입력 형식

다음 정보가 표준 입력으로 주어진다.

- 첫째 줄에는 공백으로 구분된 세 정수 N, K, Q가 주어진다. 이는 JOI 전철에는 N 개의 역이 있고, K 종류의 열차가 있으며, 질문의 개수가 Q 개라는 의미이다.
- 다음 N 개의 줄의 i 번째 $(1 \leq i \leq N)$ 줄에는 정수 L_i가 주어진다. 이는 i번 역의 중요도가 L_i라는 의미이다.
- 다음 Q 개의 줄의 k 번째 $(1 \leq k \leq Q)$ 줄에는 공백으로 구분된 두 정수 A_k, B_k가 주어진다. 이는 k 번째 승객의 출발역이 A_k번 역이고, 도착역이 B_k번 역이라는 의미이다.

출력 형식

표준 출력으로 Q 개의 줄을 출력하여라. k 번째 $(1 \leq k \leq Q)$ 줄에는 A_k번 역에서 B_k번 역으로 이동할 때, 도중에 정차하는 철도역의 최솟값을 출력하여라.

제한

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

서브태스크 1 (5 점)

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

서브태스크 2 (15 점)

- $Q \leq 50$

서브태스크 3 (25 점)

- $K \leq 20$

서브태스크 4 (55 점)

추가 제한조건이 없다.

예제

standard input	standard output
9 3 3 3 1 1 1 2 2 2 3 3 2 4 4 9 6 7	1 3 0

이 예제에서, 질문은 세 가지가 있다.

- 첫 번째 질문은 2번 역에서 4번 역까지 이동하는 것이다. 이때 2번 역에서 4번 역까지 1번 종류의 열차를 이용하면 도중에 정차하는 역은 3번 역 하나뿐이 된다.
- 두 번째 질문은 4번 역에서 9번 역까지 이동하는 것이다. 이때 우선 4번 역에서 5번 역까지 1번 종류의 열차를 이용하고, 다음에 5번 역에서 1번 역까지 2번 종류의 열차를 이용하고, 마지막으로 1번 역에서 9번 역까지 3번 종류의 열차를 이용하면 도중에 정차하는 역은 5번 역, 1번 역, 8번 역의 셋이 된다.
- 세 번째 질문은 6번 역에서 7번 역까지 이동하는 것이다. 이때 6번 역에서 7번 역까지 2번 종류의 열차를 이용하면 도중에 정차하는 역 없이 이동하는 것이 가능하다.

standard input	standard output
5 2 1 2 1 1 1 2 1 4	1

도중에 목적지가 있는 역을 지나쳐도 되는 점에 주의하여라.

standard input	standard output
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