마상시합 토너먼트

1491년 베아트리체의 결혼식을 위해서 밀란의 스포르자 공작은 다빈치에게 3일간의 마상시합(말을 타고 창 실력을 겨루는 게임)을 포함한 결혼 축제를 기획하도록 부탁하였다. 하지만 가장 인기있는 기사가 도착이 늦기에...

토너먼트

마상시합 토너먼트에 참가하는 N 명의 기사는 처음에 한 줄로 서 있고, 줄에 선 순서대로 기사들은 0 부터 N - 1 까지 번호가 붙어 있다. 마상시합의 한 *라운드*는 두 위치 S와 E를 ($0 \leq S < E \leq N - 1$) 부름으로써 시작된다. 그러면 번호가 S 이상 E 이하인 모든 기사들이 이번 라운드에서 경쟁하여 1명의 승자가 정해지게 된다. 승자는 다시 원래의 줄로 돌아가고, 패자들은 경기에서 빠지게 된다. 기사들이 0번 쪽으로 순서를 유지하면서 이동하여 원래 줄의 빈 자리들을 채운다. 결과적으로 남은 기사들의 번호는 0 부터 N - (E - S) - 1 까지가 된다. 다음 라운드도 같은 방식으로 계속되어 마지막 한명의 기사가 남을 때 까지 계속된다.

다빈치는 기사들의 실력이 모두 다르다는 것을 알고 있으며, 실력은 0 부터 N - 1 까지의 정수로 표현된다 (값이 클 수록 더 좋은 실력을 의미한다). 또한, 다빈치는 모든 C개의 라운드에서 불려질 번호 범위도 정확히 알고 있고, 각 라운드에서 가장 좋은 실력 값을 가진 기사가 반드시 이긴다고 확신한다.

늦게 온 기사

N 명의 기사들 중 N - 1 명은 이미 도착하여 줄을 서 있지만, 가장 인기있는 기사만이 아직 도착하지 않았다. 아직 도착하지 않은 기사의 실력 값은 R이다. 다빈치는 그 기사의 인기를 이용해 축제의 즐거움을 더할 목적으로, 그 기사가 승리하는 라운드의 수가 가장 많아지도록 그 기사의 위치를 정하고 싶어한다. 그 기사가 참여하지 않는 라운드는 무관하며 단지 그 기사가 참여해서 이기는 라운드의 수 만이 중요하다는 점에 주의하라.

예제

기사의 총 수 N = 5인 경우에, N - 1 명의 기사들은 이미 줄을 서 있고 각각의 실력 값은 줄에 있는 순서 대로 [1, 0, 2, 4]라고 하자. 늦게 온 기사의 실력 값은 R = 3임을 알수 있다. 토너먼트는 C = 3 라운드로 구성되며 각 라운드에서 불려지는 (S, E) 값들은 순서대로 다음과 같다고 하자: (1, 3), (0, 1), (0, 1).

다빈치가 늦게 온 기사를 첫번째 위치에 배치하게 되면 전체 기사들의 실력 값은 [3, 1, 0, 2, 4]가 된다. 첫번째 라운드는 위치 1, 2, 3에 있는 실력 값이 각각 1, 0 , 2인 기사들이 경쟁하고 실력 값 2인 기사가 승자가 된다. 이 라운드 후에 남은 기사들이 있는 줄의 실력 값은 순서대로 [3, 2, 4]와 같이 될 것이다. 다음 라운드는 실력 값 3인 기사와 실력 값 2인 기사가 (위치 번호로는 0과 1) 경쟁하고 실력 값 3인 기사가 승리하여, 남은 줄의 실력 값은 [3, 4]가

된다. 마지막 라운드의 승자는 실력 값 4인 기사일 것이다. 결과적으로, 늦게 온 기사는 단 하나의 라운드(두번째 라운드)에서만 승리하였다.

반면에, 다빈치가 늦게 온 기사를 실력 값 1인 기사와 0인 기사 사이에 배치하였다면, 최초의 줄의 실력 값들은 다음과 같을 것이다: [1, 3, 0, 2, 4]. 이 경우, 첫 라운드에 참여하는 기사들의 실력 값은 3, 0, 2와 같으며, 실력 값 3인 기사가 승리한다. 이제 남은 줄의 실력 값들은 [1, 3, 4]이며 그 다음 라운드에서 실력 값 1, 3인 기사들이 경쟁하여 역시 실력 값 3인 기사가 승리한다. 직후의 남은 줄의 실력 값들은 [3, 4]이며, 마지막 라운드에서는 실력 값 4인 기사가 승리한다. 따라서, 이 경우 늦게 온 기사는 2개의 라운드에서 승리하며, 늦게온 기사가 3회 이상 승리하는 것은 불가능하므로 이 방법이 가장 좋은 경우이다.

해야 할 일

다빈치가 바라는 바에 따라, 늦게 온 기사가 승리하는 라운드의 수가 최대가 되는 위치를 찾아내는 프로그램을 작성하라. 구체적으로, `GetBestPosition`(N, C, R, K, S, E) 함수를 작성하여야 한다.

- N 은 기사들의 총 수이다.
- C는 진행되는 라운드의 수이다 ($1 \le C \le N - 1$).
- R은 늦게 온 기사의 실력 값이다. 늦게 온 기사를 포함한 모든 기사들의 실력 값은 0 이상 N - 1 이하의 정수이며 모두 다르다. R의 값은 다른 입력 값들로 부터 계산이 가능하지만, 명시적으로 주어진다.
- K는 N - 1 개의 정수의 배열이며, 이미 줄에 서 있는 N - 1 명의 기사들의 실력 값을 줄에 서 있는 순서와 같은 순서로 저장하고 있다.
- S와 E는 크기 C인 배열들이다. 0 이상 C - 1 이하의 i 에 대해서 S[i]과 E[i]의 값은 (i + 1) 번째 라운드에 참여하는 기사들의 줄에서의 위치가 S[i] 부터 E[i] 까지 임을 의미한다. 모든 i에 대해서 S[i] < E[i] 임은 보장된다.

이 함수를 호출할 때 인자들의 값의 정확함이 보장된다. 즉, E[i]의 값은 (i + 1) 번째 라운드가 시작할 때 줄에 있는 기사들의 수 보다 작음이 보장되며, 모든 C개의 라운드가 끝난 다음에는 단 한명의 기사만이 남아 있음이 보장된다.

`GetBestPosition`(N, C, R, K, S, E) 함수는 늦게 온 기사를 배치할 수 있는 최적의 위치 P를 리턴하여야 한다 ($0 \le P \le N - 1$). 만약 최적의 위치가 여러 개일 경우는 *가장 작은 값*을 리턴하여야 한다. (늦게 온 기사의 위치는 그 기사가 배치된 이후의 줄에서의 위치를 말하며, 가장 앞의 위치를 0으로 시작하여 표현한 것이다. 다르게 표현하면, P는 늦게온 기사를 최적 위치에 배치한 후, 그 기사 앞에 있는 기사들의 수이다. 특정한 경우들을 보면, P = 0 이라는 것은 늦게 온 기사가 가장 앞에 위치한다는 뜻이며, P = N - 1 이라는 것은 늦게 온 기사가 줄의 가장 뒤에 위치한다는 뜻이다.)

서브태스크 1 [17 점]

이 서브태스크에서 $N \le 500$ 이다.

서브태스크 2 [32 점]

이 서브태스크에서 $N \leq 5{,}000$ 이다.

서브태스크 3 [51 점]

이 서브태스크에서 $N \leq 100{,}000$ 이다.

구현 세부 사항

단 하나의 파일을 제출하여야 하는데, 그 이름은 `tournament.c`, `tournament.cpp` 혹은 `tournament.pas`이다. 이 파일은 다음 선언을 사용해서 위에서 설명한 함수를 구현해야 한다.

C/C++ 프로그램

```
int GetBestPosition(int N, int C, int R, int *K, int *S, int *E);
```

Pascal 프로그램

```
function GetBestPosition(N, C, R : LongInt; var K, S, E : array of LongInt) : LongInt;
```

이 부프로그램들은 위에서 설명한 것과 같이 동작하여야 한다. 물론, 다른 부프로그램을 만들어서 내부적으로 사용할 수 있다. 제출한 코드는 입출력을 수행하거나 다른 파일에 접근하여서는 안된다.

Grader 예시

주어지는 grader의 입력은 다음 양식과 같다.

- line 1: N, C, R;
- lines 2, …, N: K[i];
- lines N + 1, …, N + C: S[i], E[i].

시간과 메모리 제한

- 시간 제한: 1 초.
- 메모리 제한: 256MB.