점프 게임 (jumpgame)

KOI 사가 출시한 *점프 게임*은 일렬로 나열된 N개의 발판을 여러 번의 점프를 통해서 통과하는 게임이다. 구체적으로, 수직선 상에 N 개의 발판이 존재하며, 각 발판들은 왼쪽에서 오른쪽으로 $0, 1, \ldots, N-1$ 의 번호가 매겨져 있다. 맨 처음 게임의 주인공은 가장 왼쪽 발판인 0번 발판에 있으며, 0의 점수로 시작한다.

모든 $0 \le i \le N-1$ 에 대해, i 번 발판 위에서 주인공은 **걷기** 혹은 **점프** 중 하나의 행동을 선택한다. 주인공이 **걷기** 를 선택한다면, 주인공은 $i+1$ 번 발판으로 이동하며, 점수의 변동은 없다. 주인공이 **점프** 를 선택한다면, 주인공은 $i+K$ 번 발판으로 이동하며, 점수가 $A[i]$ 만큼 증가한다. 이 때의 K는 미리 정해진 수이다. 게임은 주인공이 $N-1$번 발판의 오른쪽으로 가면 게임이 성공적으로 종료된다. 게임에서 이는, $N, N+1, \ldots$ 번 발판에 도달하는 것으로 처리된다 - 번호가 N 이상인 발판은 실제로 존재하지 않지만, $N-1$ 번 발판의 오른쪽이라는 의미로 간주된다. 게임의 목표는, 주인공을 적절히 조종하여 점수를 최대화하고 게임을 종료하는 것이다.

인터넷 방송을 취미로 하는 상혁이는 이따금씩 KOI 사의 *점프 게임*을 하고는 한다. 상혁이는 이 게임을 나름대로 재미있게 즐기는 편이지만, 방송의 시청자들에게는 아쉽게도 그다지 좋은 반응을 받지 못하고 있다. *점프 게임*이 시청자들에게 인기를 끌지 못 하는 이유는, 게임이 대단히 어렵고 지루하기 때문이다. 첫 번째로, 이 게임의 발판의 개수는 자그마치 10^{12} 개에 달할 수 있다. 두 번째로, KOI 사의 훌륭한 개발자들도 이 정도로 많은 발판을 모두 디자인 할 수는 없어서, 다소 단순한 방법으로 각 발판을 구성하였다. KOI 사의 개발자들은 초기 모든 $A[i]$ 를 0 으로 설정한 후, Q 번에 걸쳐서 다음과 같은 연산을 수행한다: 각 j $(0 \le j \le Q-1)$ 에 대해, 개발자들은 어떠한 구간 $0 \le L[j] \le R[j] \le N-1$ 를 골라 $A[L[j]], A[L[j]+1], A[L[j]+2], \ldots, A[R[j]]$ 을 1 씩 증가시켰다. 모든 연산을 마친 이후의 배열 A 가, 게임에서 각 발판에서 점프를 선택했을 때 얻는 점수가 된다.

컴퓨터 과학에 관한 영상을 제작하는 당신은 KOI 사의 *점프 게임*을 **최대의 점수로 종료**하는 방법에 대한 영상을 만드는 생각을 하였다. 당신은 이 영상이 상혁이의 인터넷 방송을 즐기는 팬들을 대상으로 큰 인기를 끌 것이라고 생각하고 있지만, 효율적인 알고리즘이 존재하기에는 너무나도 거대한 크기의 게임이라는 것이 걱정이다. 모든 어려움을 이겨내고, 주어진 5시간 안에 이 게임의 최고가 되어 보자.

함수 목록 및 정의

여러분은 아래 함수를 구현해야 한다.

```
long long play_game(long long N, int Q, long long K, vector<long long> L,
vector<long long> R)
```

- N: 게임에 존재하는 발판의 개수.
- Q: 개발자가 수행한 연산의 횟수.
- K: 점프 후 도달하는 다음 발판의 번호를 결정하는 인자.
- L, R: 크기가 Q인 정수 배열.
- 이 함수는 하나의 정수로 점프 게임이 종료했을 때 얻을 수 있는 점수의 최댓값을 반환한다.
- 이 함수는 매 테스트 케이스마다 단 한 번 호출된다.

제출하는 소스 코드의 어느 부분에서도 입출력 함수를 실행해서는 안 된다.

제약 조건

- $1 \le N \le 10^{12}$
- $1 \le Q \le 250\,000$
- $1 \le K \le N$
- 모든 $0 \le j \le Q-1$ 에 대해 $0 \le L[j] \le R[j] \le N-1$

부분문제

1. (6점)
    - $N \le 250\,000$
2. (2점)
    - $K = 1$
3. (13점)
    - $2K \ge N$
4. (15점)
    - $5K \ge N$
5. (16점)
    - $Q \le 500$
6. (7점)
    - $Q \le 5\,000$
7. (41점)
    - 추가적인 제약 조건이 없다.

예제 1

$N = 3$, $Q = 5$, $K = 2$, $L = [0, 0, 1, 1, 1]$, $R = [2, 2, 1, 1, 1]$인 경우를 생각해 보자.

그레이더는 다음과 같이 함수를 호출한다.

```
play_game(3, 5, 2, [0, 0, 1, 1, 1], [2, 2, 1, 1, 1])
```

Q 개의 연산을 수행한 이후 얻게 되는 배열은 $[2, 5, 2]$ 이다. 이 경우 얻을 수 있는 최고 점수는 5이다. 5의 점수를 얻는 방법 중 하나는 다음과 같다:

- 0 번 발판에서 걷기.
- 1 번 발판에서 점프 후 5 점 획득.
- $3 \geq N$ 이니 게임 종료.

고로 함수는 5를 반환하여야 한다.

예제 2

그레이더는 다음과 같이 함수를 호출한다.

```
play_game(250, 8, 50, [0, 40, 49, 50, 100, 149, 199, 75], [200, 140, 49, 150,
190, 199, 199, 249])
```

얻을 수 있는 최고 점수는 17이다. 고로 함수는 17을 반환하여야 한다.

예제 3

그레이더는 다음과 같이 함수를 호출한다.

```
play_game(250, 8, 49, [0, 40, 49, 50, 100, 149, 199, 75], [200, 140, 49, 150,
190, 199, 199, 249])
```

얻을 수 있는 최고 점수는 19이다. 고로 함수는 19를 반환하여야 한다.

예제 4

그레이더는 다음과 같이 함수를 호출한다.

```
play_game(100, 6, 50, [0, 0, 0, 20, 40, 60], [99, 70, 10, 30, 50, 70])
```

얻을 수 있는 최고 점수는 6이다. 고로 함수는 6을 반환하여야 한다.

예제 5

그레이더는 다음과 같이 함수를 호출한다. (가독성을 위해 일부 숫자 사이에 공백이 추가되었다.)

```
play_game(1 000 000 000 000, 2, 1, [0, 0], [999 999 999 999, 999 999 999 999])
```

얻을 수 있는 최고 점수는 2 000 000 000 000이다. 고로 함수는 2 000 000 000 000를 반환하여야 한다.

Sample grader

Sample grader는 아래와 같은 형식으로 입력을 받는다.

- Line 1: N Q K
- Line $2 + i$ ($0 \leq i \leq Q - 1$): $L[i]$ $R[i]$

Sample grader는 다음을 출력한다.

- Line 1: `play_game`이 반환한 값

Sample grader는 실제 채점에서 사용하는 그레이더와 다를 수 있음에 유의하라.