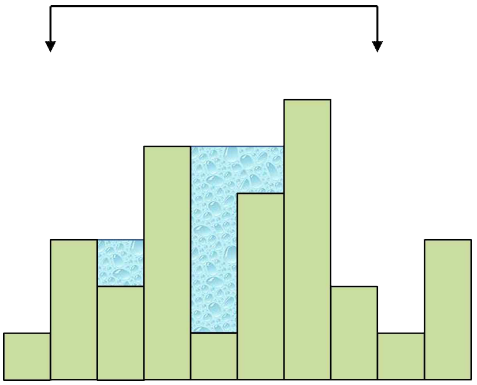
어떤 외계 행성의 지표면은 사막이 대부분이라 비가 별로 오지 않는다. 이 행성에 사는 식물은 모두 선인장이다. 비가 워낙 오지 않기 때문에, 선인장들은 서로 협력해서 물을 저장하는 방법을 가지도록 진화했다.

행성의 한 지역은 상하 방향과 좌우 방향만 존재하는 2차원 평면이다. 그 지역에 N개의 선인장이 일렬로 붙어 서 있다. 선인장의 폭은 모두 1미터이다. 선인장들의 높이는 서로 다를 수 있다. 환경미화를 위해서 S번째 선인장에서 E번째 선인장까지만 남긴 상태에서 비가 충분히 많이 내렸다. 그래서 선인장 위쪽에 물을 담을 수 있는 면적에 모두 물이 고였다고 한다. 즉, 아래 그림처럼 물이 고인다. 아래 그림에서 화살표로 표시된 범위 밖의 선인장은 없어진 것으로 생각해야 한다.



선인장들의 높이와 남아 있는 선인장들의 범위를 입력으로 받아 고인 물의 양을 면적으로 계산하는 프로그램을 작성하라. 남아 있는 선인장들의 범위는 최대 Q번 주어진다. 주어지는 범위는 항상 모든 선인장이 존재하는 초기 상태에 적용한다.

여러분은 다음 함수를 작성하여야 한다.

- `void init( int H[] )` : 최초에 한번만 호출되는 함수이다. H는 제일 왼쪽 선인장부터 순서대로 선인장의 높이를 저장한 크기 N인 배열(vector)이다. N은 선인장의 개수이다.
- `int query( int S, int E )` : S번째 선인장부터 E번째 선인장까지 남긴 상태에서 충분히 많은 비가 내렸을 때 고인 물의 양을 면적으로 리턴해야 한다. ($1 \le S \le E \le N$) 이 함수는 최대 Q번 호출된다. 모든 호출은 독립적이다. 즉, 한 호출에서 선인장이 없어진 것은 다른 호출과 무관하다.

구현 세부사항

여러분은 cactus.cpp라는 이름의 정확히 하나의 파일을 제출해야 한다. 이 파일에는 다음의 함수가 구현되어 있어야 한다.

- `void init( int H[] )`
- ~~`int`~~ `long long query( int S, int E )`

이 함수는 위에서 설명한 것과 같이 동작하여야 한다. 물론 다른 함수들을 만들어서 내부적으

로 사용할 수 있다. 제출한 코드는 입출력을 수행하거나 다른 파일에 접근하여서는 안된다.

grader 예시

주어지는 grader는 다음과 같은 형식으로 입력을 받는다.

- line 1: N, Q.
- 다음 줄: N 개의 자연수, 왼쪽부터 순서대로 선인장들의 높이이다.
- 다음 Q개의 줄: 2개의 자연수 S,E, S번째부터 E번째까지 선인장을 남긴 상태에서에 비가 내렸다는 의미이다. ($1 \leq S \leq E \leq N$)

주어진 grader는 여러분의 코드가 query() 함수에서 리턴 한 값들을 한 줄에 하나씩 출력한다.

제한 조건

- $1 \leq N \leq 500{,}000$, $1 \leq Q \leq 500{,}000$, **선인장의 높이는 1이상 10^9이하**

서브태스크 1 [13 points]

- $1 \leq N \leq 5{,}000$, $1 \leq Q \leq 5{,}000$

서브태스크 2 [12 points]

- $1 \leq N \leq 5{,}000$

서브태스크 3 [22 points]

- **선인장의 높이는 20이하**

서브태스크 4 [34 points]

- $1 \leq N \leq 100{,}000$, $1 \leq Q \leq 100{,}000$

서브태스크 5 [69 points]

- **원래 제한 조건 이외의 추가적인 제한 조건이 없음**

[입력 예]

```
10 3
1 3 2 5 1 4 6 2 1 3
2 8
2 4
7 9
```

위의 입력에서 여러분의 코드가 동작하는 방식에 따른 실행 예를 아래에 보인다.

호출	결과
init({1, 3, 2, 5, 1, 4, 6, 2, 1, 3})	초기 호출
query(2, 8)	Query 호출, 리턴 값 6
query(2, 4)	Query 호출, 리턴 값 1
query(7, 9)	Query 호출, 리턴 값 0