던전

로버트는 새로운 컴퓨터 게임을 설계하고 있다. 게임에는 한명의 영웅, n명의 적, $n+1$개의 던전이 있다. 적들은 0번부터 $n-1$번까지 번호가 붙어 있다. 던전들은 0번부터 n번까지 번호가 붙어 있다. 적 i($0 \le i \le n-1$)는 던전 i에 위치하며 그 힘은 $s[i]$이다. 던전 n에는 적이 없다.

영웅은 최초에 던전 x에서 힘 z를 가지고 시작한다. 영웅이 던전 i($0 \le i \le n-1$)에서 적 i와 대결한다고 하자. 대결의 결과로 다음 중 하나의 결과가 나온다.

- 영웅의 힘이 적의 힘 $s[i]$보다 크거나 같은 경우 영웅이 이긴다. 그 결과 영웅의 힘은 $s[i]$($s[i] \ge 1$)만큼 **증가**한다. 다음에 영웅이 들어가는 던전은 $w[i]$($w[i] > i$)이다.
- 그렇지 않은 경우 영웅이 진다. 그 결과 영웅의 힘은 $p[i]$($p[i] \ge 1$)만큼 **증가**한다. 다음에 영웅이 들어가는 던전은 $l[i]$이다.

주의: $p[i]$는 $s[i]$보다 크거나 같거나 작은 경우가 모두 가능하다. 또, $l[i]$는 i보다 크거나 같거나 작은 경우가 모두 가능하다. 대결의 승부와 무관하게 적 i는 제자리에 계속 존재하며, 힘 $s[i]$도 동일하게 유지된다.

영웅이 던전 n에 들어가면 게임이 끝난다. 이 게임이 영웅의 시작 던전과 힘이 어떤 값이든, 유한한 개수의 대결 이후에 끝난다는 것을 증명할 수 있다.

로버트는 당신에게, q번의 시뮬레이션을 돌려서 게임을 테스트해 달라고 한다. 각 시뮬레이션에 대해서 로버트는 영웅의 시작 던전 x와 시작할 때의 힘 z를 지정할 것이다. 당신이 해야 하는 일은, 각 시뮬레이션에 대해서, 게임이 끝날 때의 영웅의 힘을 계산하는 것이다.

Implementation details

다음 함수를 구현해야 한다.

```
void init(int n, int[] s, int[] p, int[] w, int[] l)
```

- n: 적의 개수.
- s, p, w, l: 길이 n인 배열들 ($0 \le i \le n-1$).
  - $s[i]$는 적 i의 힘이다. 또, 영웅이 적 i를 이겼을 때 얻는 힘의 양이다.
  - $p[i]$는 영웅이 적 i에게 졌을 때 얻는 힘이다.
  - $w[i]$는 영웅이 적 i에게 이겼을 때 그 다음에 들어가는 던전의 번호이다.
  - $l[i]$는 영웅이 적 i에게 졌을 때 그 다음에 들어가는 던전의 번호이다.
- 이 함수는 최초에 한번만 불린다. 이후에 아래에 설명된 `simulate` 호출이 이어진다.

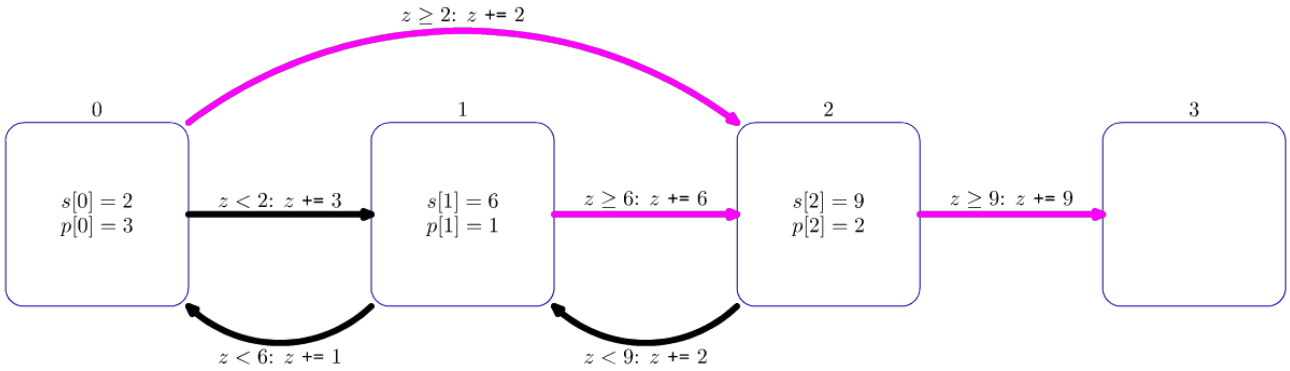
```
int64 simulate(int x, int z)
```

- x는 최초로 영웅이 들어가는 던전의 번호이다.
- z는 최초에 영웅이 가지는 힘이다.
- 이 함수는 위와 같이 시작해서 게임이 끝날 때 영웅이 가지는 힘을 리턴해야 한다.
- 이 함수는 정확히 q번 호출된다.

Example

다음 호출을 보자.

```
init(3, [2, 6, 9], [3, 1, 2], [2, 2, 3], [1, 0, 1])
```



위 그림이 이 호출의 상황을 보여준다. 각 사각형은 하나의 던전에 해당한다. 던전 0, 1, 2에 대해서, $s[i]$와 $p[i]$의 값들은 사각형 안에 표시되어 있다. 자주색 화살표는 영웅이 이겼을 때 따라가는 길을 보여준다. 검은색 화살표는 영웅이 졌을 때 따라가는 길을 보여준다.

그레이더가 `simulate(0, 1)`을 호출했다고 하자.

게임은 아래와 같이 진행된다.

던전	승부 직전의 영웅의 힘	결과
0	1	패
1	4	패
0	5	승
2	7	패
1	9	승
2	15	승
3	24	끝

따라서 함수는 24를 리턴해야 한다.

그레이더가 `simulate(2, 3)`을 호출했다고 하자.

게임은 아래와 같이 진행된다.

던전	승부 직전의 영웅의 힘	결과
2	3	패
1	5	패
0	6	승
2	8	패
1	10	승
2	16	승
3	25	끝

따라서 함수는 25를 리턴해야 한다.

Constraints

- $1 \le n \le 400\,000$
- $1 \le q \le 50\,000$
- $1 \le s[i], p[i] \le 10^7$ (모든 $0 \le i \le n-1$)
- $0 \le l[i], w[i] \le n$ (모든 $0 \le i \le n-1$)
- $w[i] > i$ (모든 $0 \le i \le n-1$)
- $0 \le x \le n-1$
- $1 \le z \le 10^7$

Subtasks

1. (11 점) $n \le 50\,000,\ q \le 100,\ s[i], p[i] \le 10\,000$ (모든 $0 \le i \le n-1$)
2. (26 점) $s[i] = p[i]$ (모든 $0 \le i \le n-1$)
3. (13 점) $n \le 50\,000$, 모든 적의 힘은 같다. 즉, 모든 $0 \le i, j \le n-1$에 대해 $s[i] = s[j]$이다.
4. (12 점) $n \le 50\,000$이고 $s[i]$로 등장하는 값들은 최대 5가지이다.
5. (27 점) $n \le 50\,000$
6. (11 점) 추가적인 제한이 없다.

Sample grader

샘플 그레이더의 입력 양식은 다음과 같다.

- line 1: $n\ \ q$
- line 2: $s[0]\ \ s[1]\ \ \ldots\ \ s[n-1]$
- line 3: $p[0]\ \ p[1]\ \ \ldots\ \ p[n-1]$
- line 4: $w[0]\ \ w[1]\ \ \ldots\ \ w[n-1]$
- line 5: $l[0]\ \ l[1]\ \ \ldots\ \ l[n-1]$
- line $6+i$ ($0 \le i \le q-1$): i번째 `simulate` 호출의 $x\ \ z$.

샘플 그레이더의 출력 양식은 다음과 같다.

- line $1 + i$ ($0 \le i \le q - 1$): i번째 `simulate` 호출의 리턴 값.