Dungeons Game

Roberto está diseñando un nuevo juego de ordenador. El juego tiene un héroe, n adversarios y $n+1$ mazmorras. Los adversarios están numerados de 0 a $n-1$ y las mazmorras están numeradas de 0 a n. El adversario i ($0 \leq i \leq n-1$ se encuentra en la mazmorra i y tiene fuerza $s[i]$. No hay adversario en la mazmorra n.

El héroe empieza entrando en la mazmorra x, con fuerza z. Cada vez que el héroe entra en una mazmorra i ($0 \leq i \leq n-1$), se enfrenta al adversario i, y luego uno de los siguientes casos ocurre:

- Si la fuerza del héroe es mayor o igual a la fuerza del adversario $s[i]$, el héroe gana. Esto **incrementa** la fuerza del héroe en $s[i]$ ($s[i] \geq 1$). En este caso el héroe entra en la mazmorra $w[i]$ a continuación ($w[i] > i$).
- Alternativamente, el héroe pierde. Esto hace que la fuerza del héroe se **incremente** en $p[i]$ ($p[i] \geq 1$). En este caso el héroe entra en la mazmorra $l[i]$ a continuación.

Nótese que $p[i]$ puede ser menor, igual o mayor que $s[i]$. También, $l[i]$ puede ser menor, igual o mayor que i. Independientemente del resultado de la confrontación, el adversario se queda en la mazmorra i, manteniendo una fuerza $s[i]$.

El juego acaba cuando el héroe entra en la mazmorra n. Uno puede demostrar que el juego acabará siempre después de un número finito de confrontaciones, independientemente de donde empiece el héroe y de su fuerza inicial.

Roberto te pide que pruebes su juego ejecutando q simulaciones. Para cada simulación, Roberto define la mazmorra de entrada x y la fuerza z. Tu tarea es averiguar para cada simulación la fuerza del héroe cuando el juego acaba.

Detalles de implementación

Tienes que implementar los siguientes procedimientos:

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

- n: número de adversarios
- s, p, w, l: vectores de longitud n. Para $0 \leq i \leq n-1$:
  - $s[i]$ es la fuerza del adversario i. También es la fuerza obtenida después de que el héroe gane al adversario i.
  - $p[i]$ es la fuerza obtenida por el héroe después de perder contra el oponente i.
  - $w[i]$ es la mazmorra donde el héroe entra después de ganar contra el oponente i.

    - $l[i]$ es la mazmorra donde el héroe entra después de perder contra el oponente i.
- Este procedimiento es llamado exactamente una sola vez, antes de cualquiera de las llamadas a `simulate` (ver más abajo).

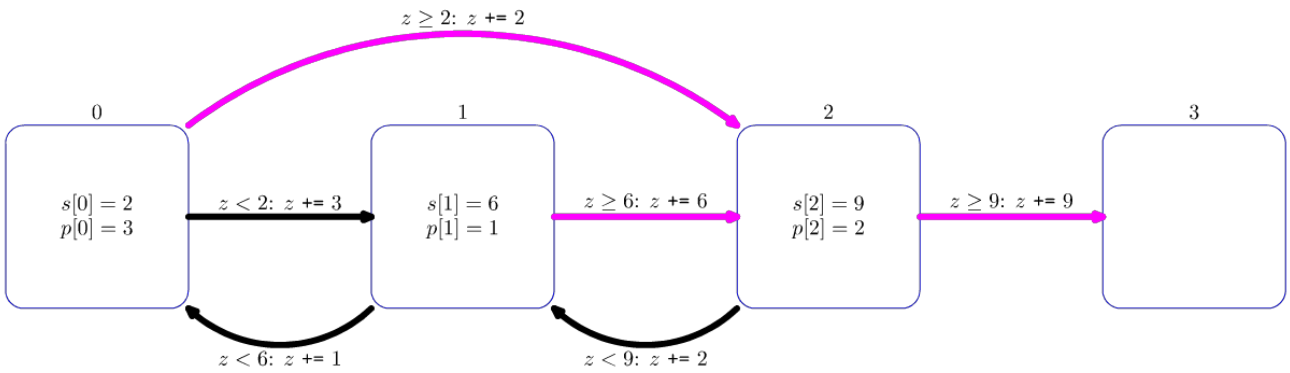
```
int64 simulate(int x, int z)
```

- x: la mazmorra en que entra inicialmente el héroe.
- z: la fuerza inicial del héroe.
- Este procedimiento debería devolver la fuerza del héroe cuando el juego acabe, asumiendo que el héroe empieza el juego entrando en la mazmorra x, con fuerza z.
- Este procedimiento se llamará exactamente q veces.

Ejemplo

Considera la siguiente llamada:

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



El diagrama anterior muestra esta llamada. Cada cuadrado muestra una mazmorra. Para las mazmorras 0, 1 y 2, los valores $s[i]$ y $p[i]$ se indican dentro. Las flechas magenta indícan dónde se moverá el héroe despues de ganar al confrontación, mientras que las flechas negras indican donde se moverá el héroe después de perder.

Digamos que el grader llama `simulate(0, 1)`.

El juego procede como sigue:

Mazmorra	Fuerza del héroe antes de la confrontación	Resultado
0	1	Pierde
1	4	Pierde
0	5	Gana
2	7	Pierde
1	9	Gana
2	15	Gana
3	24	Juego acaba

Por lo tanto, el procedimiento debe devolver 24.

Digamos que el grader llama `simulate(2, 3)`.

El juego transcurre de la siguiente manera:

Mazmorra	Fuerza del héroe antes de la confrontación	Resultado
2	3	Pierde
1	5	Pierde
0	6	Gana
2	8	Pierde
1	10	Gana
2	16	Gana
3	25	Juego acaba

Por lo tanto, el procedimiento debe devolver 25.

Restricciones

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

Subtareas

1. (11 puntos) $n \le 50\,000,\ q \le 100,\ s[i], p[i] \le 10\,000$ (para todo $0 \le i \le n-1$)

2. (26 puntos) $s[i] = p[i]$ (para todo $0 \leq i \leq n-1$)
3. (13 puntos) $n \leq 50\ 000$, todos los adversarios tienen la misma fuerza, en otras palabras, $s[i] = s[j]$ para todo $0 \leq i, j \leq n-1$.
4. (12 puntos) $n \leq 50\ 000$, hay como mucho 5 valores distintos entre todos los valores de $s[i]$.
5. (27 puntos) $n \leq 50\ 000$
6. (11 puntos) sin restricciones adicionales.

Sample grader

El sample grader lee la entrada en el siguiente formato:

- 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 $6+i$ ($0 \leq i \leq q-1$): x z para la i-ésima llamada a `simulate`.

El sample grader escribe tus respuestas en el siguiente formato:

- line $1+i$ ($0 \leq i \leq q-1$) : el valor devuelto para la i-ésima llamada a `simulate`.