Juego de calabozos

Roberto está diseñando un nuevo juego de computadora. En el juego hay un héroe, n oponentes y $n+1$ calabozos. Los oponentes se numeran desde 0 hasta $n-1$ y los calabozos se numeran desde 0 hasta n. El oponente i ($0 \leq i \leq n-1$) se encuentra ubicado en el calabozo i y tiene una fuerza $s[i]$. No hay ningún oponente en el calabozo n.

El héroe comienza ingresando al calabozo x, con fuerza z. Cada vez que el héroe ingresa a algún calabozo i ($0 \leq i \leq n-1$), se enfrenta al oponente i, y ocurre una de las siguientes cosas:

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

Notar que $p[i]$ puede ser menor, igual, o mayor que $s[i]$. Además, $l[i]$ puede ser menor, igual, o mayor que i. Sin importar el resultado del enfrentamiento, el oponente permanece en el calabozo i y mantiene una fuerza $s[i]$.

El juego termina cuando el héroe ingresa al calabozo n. Es posible demostrar que el juego termina luego de un número finito de enfrentamientos, sin importar ni el calabozo inicial ni la fuerza inicial del héroe.

Roberto te ha pedido que pruebes su juego ejecutando q simulaciones. Para cada simulación, Roberto define un calabozo inicial x y una fuerza inicial z. Tu tarea consiste en calcular, para cada simulación, la fuerza del héroe cuando termina el juego.

Detalles de implementación

Debes implementar las siguientes funciones:

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

- n: cantidad de oponentes.
- s, p, w, l: arreglos de longitud n. Para $0 \leq i \leq n-1$:
  - $s[i]$ es la fuerza del oponente i. También es la fuerza que gana el héroe luego de vencer al oponente i.
  - $p[i]$ es la fuerza que gana el héroe luego de perder contra el oponente i.
  - $w[i]$ es el calabozo al cual ingresa el héroe luego de vencer al oponente i.
  - $l[i]$ es el calabozo al cual ingresa el héroe luego de perder contra el oponente i.

- Esta función es llamada exactamente una vez, antes de realizar cualquiera llamada a `simulate` (ver más abajo).

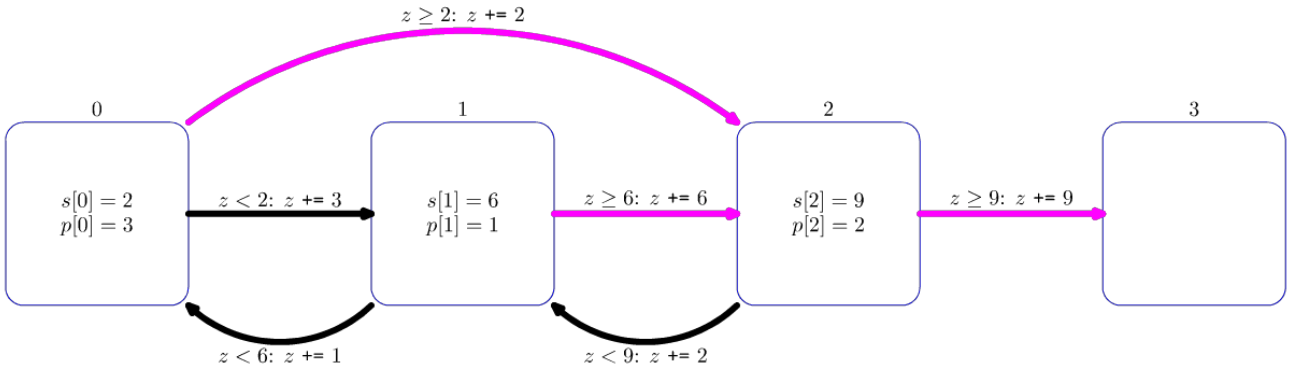
```
int64 simulate(int x, int z)
```

- x: el calabozo por el cual el héroe ingresa al comienzo.
- z: la fuerza inicial del héroe.
- Esta función debe retornar la fuerza del héroe al finalizar el juego, asumiendo que el héroe inicial el juego ingresando por el calabozo x, con una fuerza z.
- La función se llama exactamente q veces.

Ejemplo

Considera la siguiente llamada:

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



El diagrama de arriba ilustra esta llamada. Cada cuadrado muestra un calabozo. Para los calabozos 0, 1 y 2, los valores $s[i]$ y $p[i]$ se indican dentro de los cuadrados. Las flechas magenta indican a dónde se mueve el héroe luego de ganar un enfrentamiento, mientras que las flechas negras indican a dónde se mueve luego de perder.

Supongamos que el evaluador realiza la llamada `simulate(0, 1)`.

El juego procede de la siguiente manera:

Calabozo	Fuerza del héroe antes del enfrentamiento	Resultado
0	1	Derrota
1	4	Derrota
0	5	Victoria
2	7	Derrota
1	9	Victoria
2	15	Victoria
3	24	Fin del juego

De esta manera, la función debe retornar 24.

Supongamos que el evaluador realiza la llamada `simulate(2, 3)`.

El juego procede de la siguiente manera:

Calabozo	Fuerza del héroe antes del enfrentamiento	Resultado
2	3	Derrota
1	5	Derrota
0	6	Victoria
2	8	Derrota
1	10	Victoria
2	16	Victoria
3	25	Fin del juego

De esta manera, la función debe retornar 25.

Restricciones

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

Subtareas

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

2. (26 puntos) $s[i] = p[i]$ (para todo $0 \leq i \leq n-1$)
3. (13 puntos) $n \leq 50\ 000$, todos los oponentes tienen la misma fuerza, es decir, $s[i] = s[j]$ para todo $0 \leq i, j \leq n-1$.
4. (12 puntos) $n \leq 50\ 000$, existen a lo sumo 5 valores diferentes entre todos los valores $s[i]$.
5. (27 puntos) $n \leq 50\ 000$
6. (11 puntos) Sin más restricción.

Evaluador local

El evaluador local lee la entrada con el siguiente formato:

- línea 1: n q
- línea 2: $s[0]$ $s[1]$ $\ldots$ $s[n-1]$
- línea 3: $p[0]$ $p[1]$ $\ldots$ $p[n-1]$
- línea 4: $w[0]$ $w[1]$ $\ldots$ $w[n-1]$
- línea 5: $l[0]$ $l[1]$ $\ldots$ $l[n-1]$
- línea $6+i$ ($0 \leq i \leq q-1$): x z para la i-ésima llamada a `simulate`.

El evaluador local escribe la salida con el siguiente formato:

- línea $1+i$ ($0 \leq i \leq q-1$) : el valor retornado por la i-ésima llamada a `simulate`.