Juego de Calabozos

Robert está diseñando un nuevo videojuego. El juego involucra a un héroe, n oponentes y $n+1$ calabozos. Los oponentes están numerados de 0 a $n-1$ y los calabozos están numerados de 0 a n. El oponente i ($0 \leq i \leq n-1$) está ubicado en el calabozo i y tiene fuerza $s[i]$. No existe oponente en el calabozo n.

El héroe inicia entrando en el calabozo x, con fuerza z. Cada vez que el héroe entra en algún calabozo i ($0 \leq i \leq n-1$), se enfrenta al oponente i, y ocurre una de las siguientes situaciones:

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

Nótese que $p[i]$ puede ser menor que, igual a, o mayor que $s[i]$. Además, $l[i]$ puede ser menor que, igual a o mayor que i. Independientemente del resultado del enfrentamiento, el oponente permanece en el calabozo i y mantiene su fuerza $s[i]$.

El juego termina cuando el héroe entra al calabozo n. Se puede demostrar que el juego termina después de un numero finito de enfrentamientos, independientemente del calabozo y fuerza inicial del héroe.

Robert te pidió que probaras su juego ejecutando q simulaciones. Para cada simulación, Robert define un calabozo inicial x y una fuerza inicial z. Tu tarea es averiguar, para cada simulación, la fuerza del héroe cuando termina el juego.

Detalles de implementación

Se debe 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 tamaño n. Para $0 \leq i \leq n-1$:
  - $s[i]$ es la fuerza del oponente i. También es la fuerza que gana el héroe después de vencer al oponente i.
  - $p[i]$ es la fuerza que gana el héroe después de perder contra el oponente i.
  - $w[i]$ es el calabozo al que entra el héroe después de vencer al oponente i.
  - $l[i]$ es el calabozo al que entra el héroe después de perder contra el oponente i.

- Esta función es llamada exactamente una vez, antes de cualquier llamada a `simulate` (véase abajo).

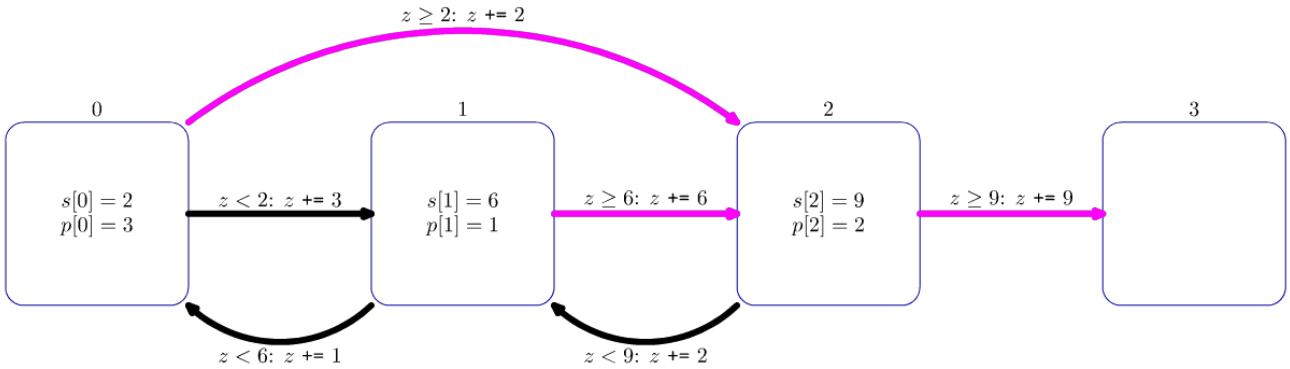
```
int64 simulate(int x, int z)
```

- x: el calabozo al que el héroe entra primero.
- z: la fuerza inicial del héroe.
- Esta función debe devolver la fuerza del héroe cuando el juego termina, asumiendo que el héroe inicia el juego entrando al calabozo x, con fuerza z.
- La función es llamada exactamente q veces.

Ejemplo

Considere 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 representa un calabozo. Para los calabozos 0, 1 y 2, los valores $s[i]$ y $p[i]$ están indicados dentro de los cuadrados. Las flechas magenta indican donde se mueve el héroe después de ganar un enfrentamiento, mientras que las flechas negras indican donde se mueve el héroe después de perder.

Digamos que el evaluador hace la llamada `simulate(0, 1)`.

El juego procede de la siguiente manera:

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

Como tal, la función debe retornar 24.

Digamos que el evaluador hace la llamada `simulate(2, 3)`.

El juego procede de la siguiente manera:

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

Como tal, 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, en otras palabras, $s[i] = s[j]$ para todo $0 \leq i, j \leq n-1$.
4. (12 puntos) $n \leq 50\ 000$, a lo sumo hay 5 valores distintos entre todos los valores $s[i]$.
5. (27 puntos) $n \leq 50\ 000$
6. (11 puntos) Sin restricciones adicionales.

Evaluador de ejemplo

El evaluador de ejemplo lee la entrada en 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 de ejemplo imprime tus respuestas en el siguiente formato:

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