Juego de Calabozos

Robert está diseñando un nuevo juego de computadora. El juego incluye a un heroe, n enemigos y $n+1$ calabozos. Los enemigos estan numerados del 0 al $n-1$ y los calabozos están numerados del 0 al n. El enemigo i ($0 \leq i \leq n-1$) se encuentra en al calabozo i y tiene fuerza $s[i]$. No hay enemigos en el calabozo n.

El heroe empieza entrando al calabozo x,con fuerza z. Cada vez que el heroe entra a un calabozo i ($0 \leq i \leq n-1$), se enfrenta al enemigo i, y ocurre uno de los siguientes:

- Si la fuerza del heroe es mayor o igual a la fuerza del enemigo, representada por $s[i]$, el heroe gana. Esto causa que la fuerza del heroe **crezca** por $s[i]$ ($s[i] \geq 1$). En este caso el heroe entra al calabozo $w[i]$ después ($w[i] > i$).
- De otra manera el heroe pierde. Esto causa que la fuerza del heroe **crezca** por $p[i]$ ($p[i] \geq 1$). En este caso el heroe entra al calabozo $l[i]$ después.

Nota que $p[i]$ puede ser menor que, igual que, o mayor que $s[i]$. También, $l[i]$ puede ser menor que, igual que, o mayor que i. Sin importar el resultado del enfrentamiento, el enemigo se queda en el calabozo i y mantiene fuerza $s[i]$.

El juego acaba cuando el heroe entra al calabozo n. Uno puede demostrar que el juego acaba después de una cantidad finita de enfrentamientos, sin importar en que calabozo haya empezado el heroe, o su fuerza inicial.

Robert te ha pedido que pruebes su juego corriendo q simulaciones. Para cada simulación, Robert define un calabozo inicial x y una fuerza inicial z. Tu tarea es encontrar, para cada simulación, la fuerza del heroe al acabar el juego.

Detalles de implementación

Debes implementar los siguientes procedimientos:

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

- n: número de enemigos,
- s, p, w, l: arreglos de tamaño n. Para $0 \leq i \leq n-1$:
  - $s[i]$ es la fuerza del enemigo i. Tambien es la fuerza ganada por el heroe despues de ganar contra el enemigo i.
  - $p[i]$ es la fuerza ganada por el heroe después de perder contra el enemigo i.
  - $w[i]$ es el calabozo al que el heroe entra después de ganar contra el enemigo i.
  - $l[i]$ es el calabozo al que el heroe entra después de perder contra el enemigo i.

- Este procedimiento es llamado exactamente una vez, antes de cualquier llamada a `simulate` (ver abajo).

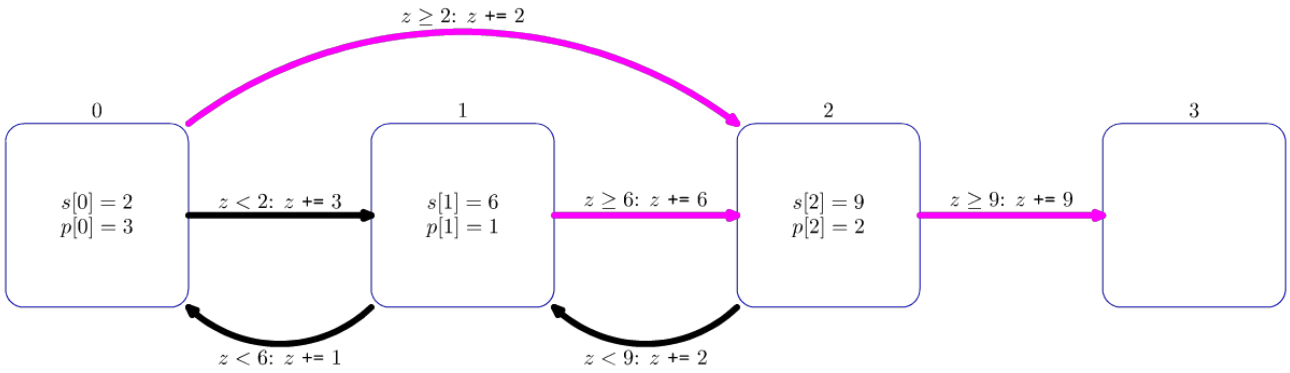
```
int64 simulate(int x, int z)
```

- x: el calabozo al que el heroe entra primero.
- z: la fuerza inicial del heroe.
- Este procedimiento debe regresar la fuerza del heroe cuando el juego acabe, asumiendo que el heroe empieza el juego al entrar al calabozo x, teniendo fuerza z.
- El procedimiento es llamado 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 $p[i]$ están indicados dentro de los cuadrado. Las flechas magenta indican a donde se mueve el heroe después de ganar un enfrentamiento, mientras que las negras indican a donde se mueve el heroe depués de perder.

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

El juego procede de la siguiente manera:

Calabozo	Fuerza del heroe antes del enfrentamiento	Resultado
0	1	Pierde
1	4	Pierde
0	5	Gana
2	7	Pierde
1	9	Gana
2	15	Gana
3	24	Acaba el juego

Por lo tanto, el procedimiento debe regresar 24.

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

El juego procede de la siguiente manera:

Calabozo	Fuerza del heroe antes del enfrentamiento	Resultado
2	3	Pierde
1	5	Pierde
0	6	Gana
2	8	Pierde
1	10	Gana
2	16	Gana
3	25	Acaba el juego

Por lo tanto, el procedimiento debe regresar 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 enemigos 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 más hay 5 valores distintos entre los $s[i]$.
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 tu respuesta en el siguiente formato:

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