Dungeons Game

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

El héroe comienza entrando al calabozo x, con fuerza z. Cada vez que el héroe entra a cualquier calazobo i ($0 \leq i \leq n-1$), confronta al oponente i, y pasa uno de los siguientes casos:

- Si la fuerza del héroe is mayor o igual a la fuerza del oponente $s[i]$, el héroe gana. Esto causa que la fuerza del héroe **aumente** $s[i]$ ($s[i] \geq 1$) puntos. En este caso el héroe entra al calabozo $w[i]$ después ($w[i] > i$)
- De otra manera, el héroe pierde. Esto causa que la fuerza del héroe **aumente** $s[i]$ ($s[i] \geq 1$) puntos. En este caso el héroe entra al calazobo $l[i]$ después.

Toma en cuenta que $p[i]$ puede ser menor, igual, o mayor que $s[i]$. Adicionalmente, $l[i]$ puede ser menor, igual o mayor que i. Sin importar el resultado de la confrontación, el oponente permanece en el calabozo i y mantiene una fuerza de $s[i]$.

El juego termina cuando el héroe entra al dungeon n. Se puede demostrar que el juego termina luego de un número finito de confrontaciones, indiferentemente del calabozo donde comienza el héroe y su fuerza.

Robert the pide que pruebes su juego corriendo q simulaciones. Por cada simulación, Robert define un calabozo inicial x y una fuerza inicial z. Tu tarea es conseguir, para cada simulación, la fuerza del héroe cuando el juego termina.

Detalles de implementación

Tu debes implementar los siguientes procedimientos:

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

- n: número 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 obtenida por el héroe luego de ganarle al oponente i.
  - $p[i]$ es la fuerza obtenida por el héroe luego de perder contra el oponente i.
  - $w[i]$ es el calabozo al que entra el héroe luego de ganarle al oponente i.
  - $l[i]$ es el calabozo al que entra el héroe luego de perder contra el oponente i.

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

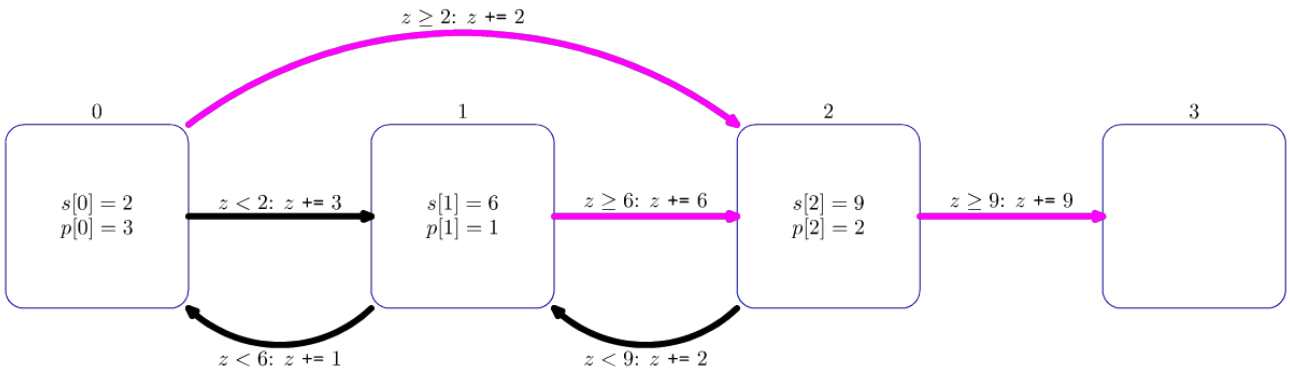
```
int64 simulate(int x, int z)
```

- x: el calabozo al que el héroe entra primero.
- z: la fuerza con la que comienza el héroe.
- Este procedimiento debe retornar la fuerza del héroe cuando termina el juego, asumiendo que el héroe comienza el juego entrando al calazobo x, teniendo una fuerza de z.
- Este procedimiento es llamado exáctamente q veces.

Ejemplo

Considera la siguiente llamada:

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



El diagrama anterior ilustra esta llamada. Cada cuadro muestra un calabozo. Para los calabozos 0, 1 and 2, los valores $s[i]$ y $p[i]$ están indicados dentro de los cuadros. Las flechas magenta indican a donde se mueve el héroe luego de ganar una confrontación, mientras que las flechas negras indican donde se mueve el héroe luego de perder.

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

El juego procede como sigue:

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

Por lo tanto, el procedimiento debe retornar 24.

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

El juego procede como sigue:

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

Por lo tanto, el procedimiento 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) $n \leq 50\ 000$, hay a lo sumo 5 valores únicos entre los valores de $s[i]$.
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`.