Kerker

Robert hat ein neues Computerspiel entworfen. Das Spiel involviert eine Heldin, n Gegner und einen Kerker mit $n+1$ Räumen. Die Gegner sind von 0 bis $n-1$ nummeriert; die Räume sind von 0 bis n nummeriert. Gegner i ($0 \leq i \leq n-1$) befindet sich in Raum i und besitzt eine Stärke von $s[i]$. In Raum n befindet sich kein Gegner.

Die Heldin beginnt in Raum x (dem Startraum) mit einer Stärke von z (der Anfangsstärke). Jedes Mal, wenn die Heldin einen Raum i ($0 \leq i \leq n-1$) betritt, sieht sie sich mit Gegner i konfrontiert und duelliert sich mit diesem. Nun passiert folgendes:

- Wenn die Stärke der Heldin größer oder gleich der Stärke $s[i]$ des Gegners ist, so gewinnt die Heldin das Duell. Weiterhin **erhöht** sich die Stärke der Heldin um $s[i]$ ($s[i] \geq 1$). In diesem Fall betritt die Heldin als Nächstes den Raum $w[i]$ ($w[i] > i$).
- Andernfalls verliert die Heldin das Duell. Weiterhin **erhöht** sich die Stärke der Heldin um $p[i]$ ($p[i] \geq 1$). In diesem Fall betritt die Heldin als Nächstes den Raum $l[i]$.

Beachte, dass $p[i]$ sowohl kleiner, gleich, als auch größer als $s[i]$ sein kann. Weiterhin kann $l[i]$ sowohl kleiner, gleich, als auch größer als i sein. Wie auch immer das Duell mit dem Gegner verlief, verbleibt der Gegner in Raum i und behält seine Stärke $s[i]$.

Das Spiel endet, sobald die Heldin Raum n betritt. Man kann zeigen, dass das Spiel immer nach einer endlichen Anzahl von Duellen endet, unabhängig von Startraum und Anfangsstärke der Heldin.

Robert bittet dich nun um Mithilfe beim Testen seines Spiels, indem du q Simulationen für ihn durchführst. Für jede Simulation bestimmt Robert einen Startraum x und eine Anfangsstärke z. Deine Aufgabe ist es nun, herauszufinden, welche Stärke die Heldin bei Spielende besitzt.

Implementierungsdetails

Du sollst die folgende Funktion implementieren:

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

- n: die Anzahl der Gegner.
- s, p, w, l: Arrays der Länge n. Für $0 \leq i \leq n-1$:
  - $s[i]$ beschreibt die Stärke von Gegner i. Dies ist auch die Stärke, welche die Heldin bei einem Sieg über den Gegner i hinzugewinnt.
  - $p[i]$ beschreibt die Stärke, welche die Heldin bei einer Niederlage gegen Gegner i hinzugewinnt.
  - $w[i]$ beschreibt den Raum, welchen die Heldin nach einem Sieg über Gegner i betritt.

    - $l[i]$ beschreibt den Raum, welchen die Heldin nach einer Niederlage gegen Gegner i betritt.
- Diese Funktion wird genau einmal aufgerufen und dies vor allen Aufrufen der Funktion `simulate` (siehe unten).

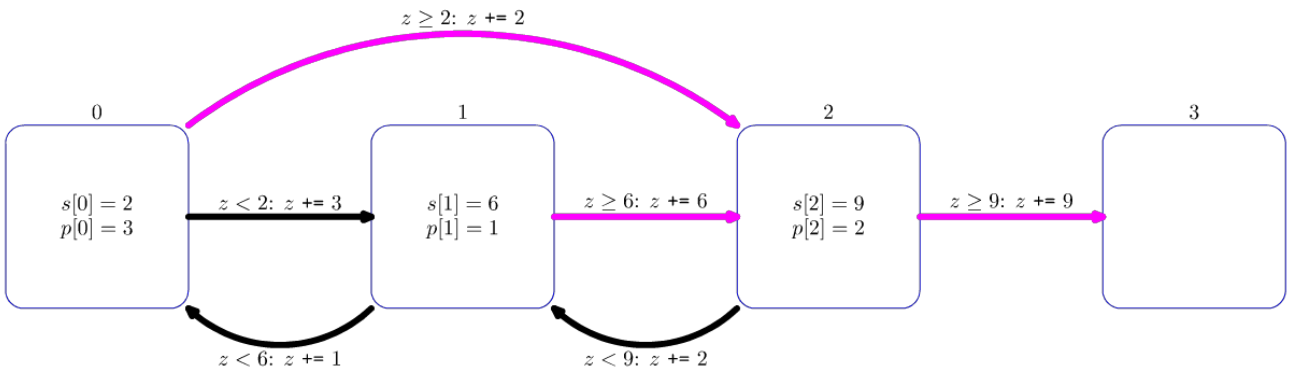
```
int64 simulate(int x, int z)
```

- x: der Startraum der Heldin.
- z: die Anfangsstärke der Heldin.
- Diese Funktion soll die Stärke der Heldin bei Spielende zurückgeben, wenn die Heldin in Raum x startet und dabei eine Stärke von z hat.
- Diese Funktion wird genau q Mal aufgerufen.

Beispiel

Betrachte den folgenden Aufruf:

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



Die obige Abbildung veranschaulicht den Funktionsaufruf. Jedes Quadrat zeigt einen Raum des Kerkers. Die Werte $s[i]$ und $p[i]$ der Räume 0, 1 und 2 stehen innerhalb der Quadrate. Magentafarbene Pfeile verdeutlichen die Bewegungen der Heldin nach einem gewonnenen Duell; schwarze Pfeile verdeutlichen die Bewegungen der Heldin nach einem verlorenen Duell.

Ruft der Grader nun `simulate(0, 1)` auf, so verläuft das Spiel wie folgt:

Kerkerraum	Stärke der Heldin vor dem Duell	Ergebnis
0	1	Niederlage
1	4	Niederlage
0	5	Sieg
2	7	Niederlage
1	9	Sieg
2	15	Sieg
3	24	Spielende

Folglich soll die Funktion 24 zurückgeben.

Ruft der Grader nun `simulate(2, 3)` auf, so verläuft das Spiel wie folgt:

Kerkerraum	Stärke der Heldin vor dem Duell	Ergebnis
2	3	Niederlage
1	5	Niederlage
0	6	Sieg
2	8	Niederlage
1	10	Sieg
2	16	Sieg
3	25	Spielende

Folglich soll die Funktion 25 zurückgeben.

Beschränkungen

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

Subtasks

1. (11 Punkte) $n \leq 50\,000,\ q \leq 100,\ s[i], p[i] \leq 10\,000$ (für alle $0 \leq i \leq n-1$)
2. (26 Punkte) $s[i] = p[i]$ (für alle $0 \leq i \leq n-1$)

3. (13 Punkte) $n \leq 50\ 000$, alle Gegner haben die gleiche Stärke, es gilt also $s[i] = s[j]$ für alle $0 \leq i, j \leq n - 1$.
4. (12 Punkte) $n \leq 50\ 000$, es gibt höchstens 5 verschiedene Werte unter allen Werten von $s[i]$.
5. (27 Punkte) $n \leq 50\ 000$
6. (11 Punkte) Keine weiteren Beschränkungen.

Beispiel-Grader

Der Beispiel-Grader liest die Eingabe in folgendem Format:

- Zeile 1: n q
- Zeile 2: $s[0]$ $s[1]$ $\ldots$ $s[n-1]$
- Zeile 3: $p[0]$ $p[1]$ $\ldots$ $p[n-1]$
- Zeile 4: $w[0]$ $w[1]$ $\ldots$ $w[n-1]$
- Zeile 5: $l[0]$ $l[1]$ $\ldots$ $l[n-1]$
- Zeile $6 + i$ ($0 \leq i \leq q - 1$): x z für den i-ten Aufruf von `simulate`.

Der Beispiel-Grader gibt deine Antworten in folgendem Format aus:

- Zeile $1 + i$ ($0 \leq i \leq q - 1$) : den Rückgabewert des i-ten Aufrufs von `simulate`.