DMOPC '15 Contest 5 P5 - World Line Convergence

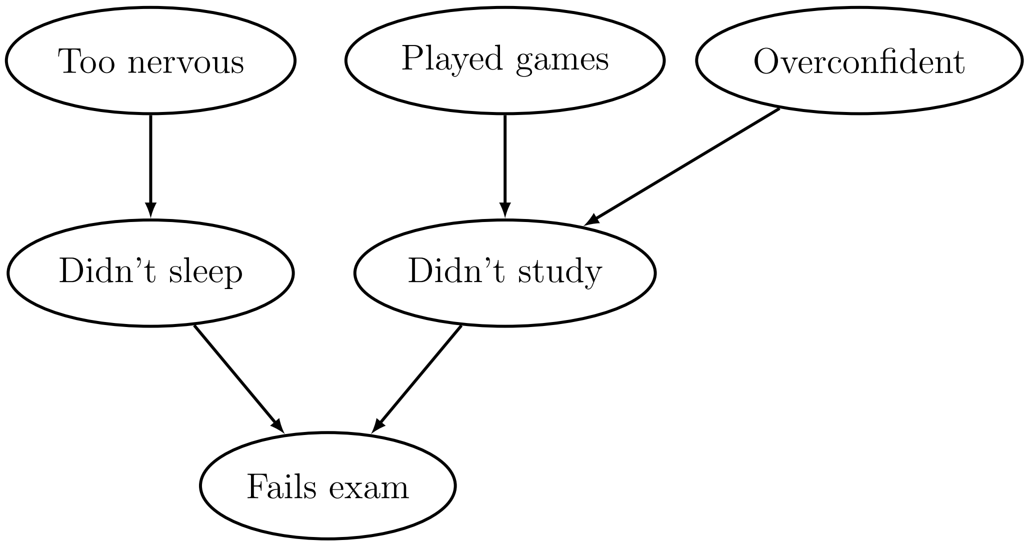
Time limit: 1.0s
Haskell: 2.5s
PyPy 2: 2.5s
PyPy 3: 2.5s

Memory limit: 128M
Haskell: 256M
PyPy 2: 256M
PyPy 3: 256M

The world of Steins;Gate is composed of many world lines in which different events occur as the results of distinct actions. However, there exists a phenomenon known as *World Line Convergence* in which multiple actions lead to the same ending, therefore leading to the convergence of multiple world lines. The place where they meet is known as a *point of convergence* or POC.

The Steins;Gate universe may therefore be seen as a tree structure where the world lines are represented by edges and the POCs by vertices. The world lines are initially created through POCs that represent the leaves of the tree and gradually converge at other POCs until all world lines have been merged into a single one at the final POC – the True End, or the root of the tree.

For example, the following tree may be representative of Okabe's life in high school. The True End in this set of world lines is Okabe's failure on his exam.



Okabe is time-leaping (you can think of this as teleportation) among N points of convergence. These POCs are labelled by the numbers 1 through N. As he travels through time, he is carrying a Divergence Meter with him, which shows the *Divergence Factor* of the POC in which he is currently located. The Divergence Factor of a POC is calculated in the following manner:

- Each point of convergence has an initial Divergence Factor of 1.
- Due to the rippling effects of time travel, whenever Okabe visits a POC, its Divergence Factor is passed on through its world line. In other words, **the Divergence Factor of the current POC is added to the Divergence Factors of all the POCs that are located between the current POC and the True End, including the True End itself.**

Okabe will visit each POC **exactly once**. Given the order in which Okabe time-leaps to the POCs, calculate the Divergence Factor of each POC at the time that Okabe visits it. Since these numbers can be quite large, you should output them modulo $1\,000\,000\,007$.

Constraints

Subtask 1 [10%]

$1 \le N \le 5\,000$

Subtask 2 [90%]

$1 \le N \le 200\,000$

Input Specification

The first line of input will contain N, the number of points of convergence.

The second line of input will contain N space-separated integers describing the tree structure. The i^{th} number is the label of the i^{th} POC's parent in the tree. For the root of the tree, the parent's label will be 0.

The third line of input will contain a permutation of the numbers from 1 to N, denoting the order in which Okabe visits the POCs.

Output Specification

Output N space-separated integers on one line, where the i^{th} number is the Divergence Factor of the POC labelled i at the time that Okabe visits it, modulo $1\,000\,000\,007$.

Sample Input

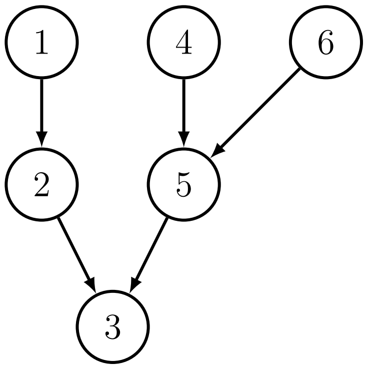
```
6
2 3 0 5 3 5
1 6 2 4 3 5
```

Sample Output

```
1 2 6 1 3 1
```

Explanation

The tree structure is identical to the previous diagram, the labels are as follows:



Let's keep track of each POC's Divergence Factor as Okabe visits them in order:

- initially, all POCs have Divergence Factors of 1: $[1, 1, 1, 1, 1, 1]$
- 1: $[1, 2, 2, 1, 1, 1]$
- 6: $[1, 2, 3, 1, 2, 1]$
- 2: $[1, 2, 5, 1, 2, 1]$
- 4: $[1, 2, 6, 1, 3, 1]$
- 3: $[1, 2, 6, 1, 3, 1]$
- 5: $[1, 2, 9, 1, 3, 1]$