Table Activation

Objective
Give extra practice with trees in C.
Give practice with heaps/min queues in C.

Story
Some concepts for this problem are taken from the Exciting Tournament homework. Please reread the story for more details.

The tournament has started; all the players have lined up, but none of the tables are lighting up. It looks like someone might have missed out on giving your team the schedule. You will just have to make it with a program on the fly.

You have the initial tournament tree structure. You will need to activate the tables in the order that creates the same tree structure. However, you decided that you will always choose to activate the table with the lowest excitement (least difference in skills) to "save the best for last". If two tables will have the same least excitement choose the one with the lowest player skill.

Problem
Given the topology of the tree and the player skill array, determine the table activation order.

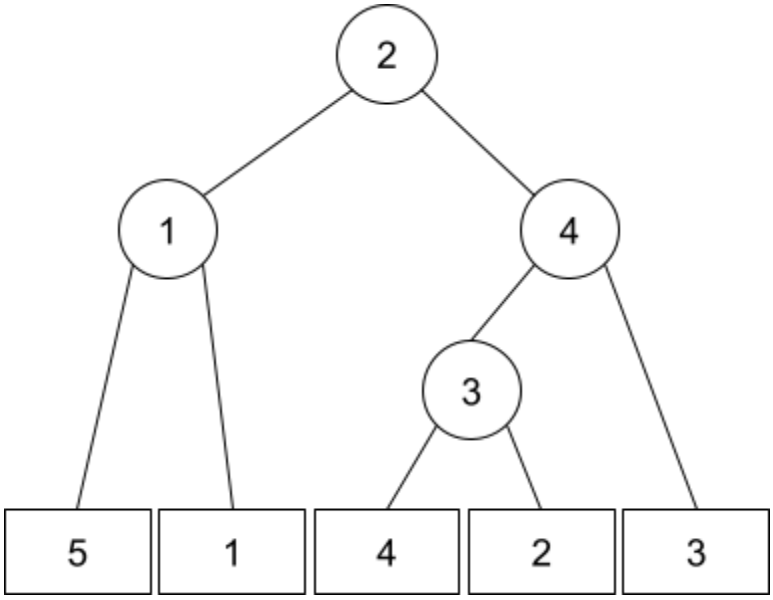
Input
Input will begin with a line containing 1 integer, n ($1 \leq n \leq 500{,}000$), representing the number of players in the tournament. The following line will contain n - 1 integers, p_1, p_2, …, p_n, ($p_i = -1$ OR $1 \leq p_i \leq n - 1$) the i-th of which (p_i) represent the table ID that the victor of table i plays in. If p_i is -1, then the table is the championship table. It is guaranteed that the table tree given will be constructable through some series of table activations. single name to process. The following and last line of input will contain n distinct positive integers, s_1, s_2, …, s_n, ($0 \leq s_i \leq 1{,}000{,}000{,}000$), representing the skill of each player in the order of their initial positions.

Output
Output should contain n - 1 lines. Each line will contain an integer representing the table that should be activated next to ensure that the least excitement is generated.

Sample Input	Sample Output
5 2 -1 4 2 5 1 4 2 3	3 4 1 2
6 -1 1 5 3 2 3 19 10 20 13 6	4 3 5 2 1

Explanation

Case 1



We have the above tournament tree. There are many possible activation orders that could produce the above tree. We have [1, 3, 4, 2] OR [3, 1, 4, 2] OR [3, 4, 1, 2].

The first order would give the excitement of
4, 2, 1, 1

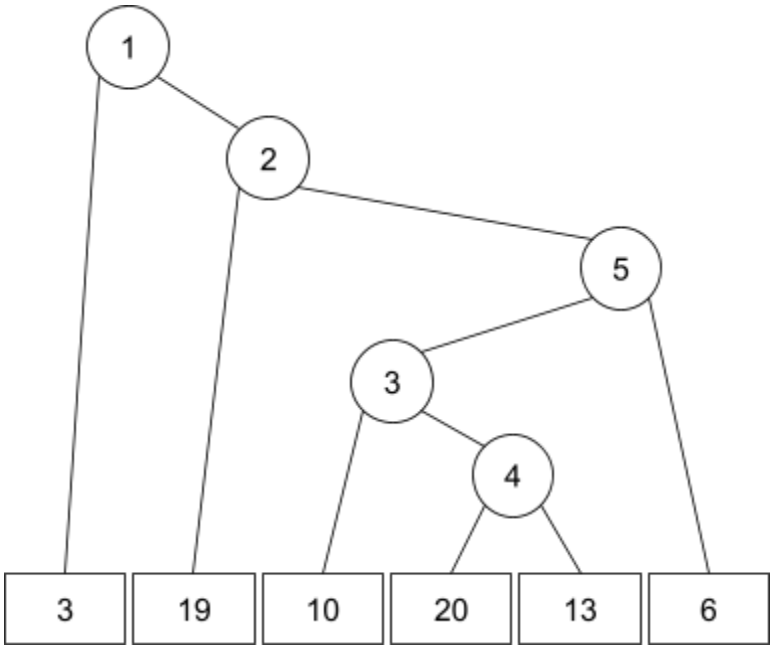
The second order would give the excitement of
2, 4, 1, 1

The last would give the excitement of
2, 1, 4, 1

The last order is the “best”, because we don’t use 4 until later.

In the actual problem we only care that the table being activated will have the least excitement generated. For this reason the first order is bad, because we can allow for an excitement of 2. The second is bad, because we can allow for 1 excitement in the 2nd match.

Case 2



We have the above tournament tree. There is only 1 possible activation order [4, 3, 5, 2, 1]. Not many options here :\

The excitements generated are the following

7, 10, 14, 1, 17

Hints

Possible Activations: Not all tables can be activated at any time. We NEED the tree structure to match the input one. To do this we need to ensure that the two players that will compete in the table have done ALL their earlier tables first. This execution order must be done this way recursively.

We can reduce the problem to ensuring that the two immediate tables (if they exist) are activated prior to the current table. We can keep track (for each table) which of their two immediate tables have been completed.

Get Least Excitement First: We need to output the table that generates the least amount of excitement. We should “store” all the tables that can be activated. If this information is stored in a way were a structure to have access to the least excitement, then we can find quickly the best table to activate.

Grading Criteria

- Read/Write from/to **standard** input/output (e.g. scanf/printf and no FILE *)
    - 10 points
- Good comments, whitespace, and variable names
    - 15 points
- Store the tables that have not been activated
    - 10 points
- Keep track of the number of incoming tables a table needs to have activated before it can be activated itself
    - 5 points
- Use the skill of the players to tie break tables with the same excitement.
    - 10 points
- Programs will be tested on 10 cases
    - 5 points each

There will be a 10 point penalty for having extra output (e.g. input prompts).

No points will be awarded to programs that do not compile using "gcc -std=gnu11 -lm".

Sometime a requested technique will be given, and solutions without the requested technique will have their maximum points total reduced. For this problem use a binary heap. ***Without this programs will earn at most 50 points!***

Any case that causes a program to return a non-zero return code will be treated as wrong. Additionally, any case that takes longer than the maximum allowed time (the max of {5 times my solutions time, 10 seconds}) will also be treated as wrong.

No partial credit will be awarded for an incorrect case.