Lockout

Problem ID: lockout

Mario worlds

Mario and Luigi are speedrunning Super Mario Odyssey to determine who is the better speedrunner. In Super Mario Odyssey, there is a collection of N worlds, where world i has a_i coins and takes t_i seconds to clear. If Mario or Luigi are the first to enter a world, then they will collect all a_i coins from it, leaving none for the other. If Mario and Luigi both enter the same world at the same time, there is a $50/50$ chance for who will receive all of the coins (in other words, Mario has a 50% chance of receiving all the coins, and Luigi does as well). Note that neither player knows which worlds have coins left at any moment. Once they enter a world, they must spend t_i seconds clearing it regardless of whether it has coins.

Fortunately for Mario, Luigi's planning was leaked and the ordering of worlds he will enter is known. That is, Luigi will begin with the first world on his list, then move to the next world, and so on. Knowing this, Mario is asking you for help to determine the optimal ordering of worlds to enter. Please determine which permutation of worlds to enter to maximize his chances of beating Luigi. Mario wins if he has strictly more coins than Luigi.

Input

The first line of input contains one integer N ($1 \leq N \leq 100,000$), representing the number of worlds. The second line contains N space-separated integers $a_1, a_2, \ldots, a_N$ ($1 \leq a_i \leq 10^9$), representing the number of coins in each world. The third line contains N space-separated integers $t_1, t_2, \ldots, t_N$ ($1 \leq t_i \leq 10^9$), representing the time it takes to clear each world. The fourth line contains N space-separated integers $p_1, p_2, \ldots, p_N$ ($1 \leq t_i \leq N$), representing Luigi's ordering of worlds: Luigi will explore world p_1 first, then world p_2, and so on. It is guaranteed that p is a permutation of 1 to N.

Output

Output one line consisting of a single double r, the maximum chance for Mario to win. This probability should not exceed an absolute error of more than 10^{-5}. On the next line, output N spaced integers, denoting the optimal permutation. If there is more than one correct permutation, then print any of them.

Sample Input 1

```
3
10 10 10
2 3 1
1 2 3
```

Sample Output 1

```
1
3 2 1
```

Sample Input 2

```
1
10
5
1
```

Sample Output 2

```
0.5
1
```