Problem A. **Perfect Game**

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

| Small input 3 points | Solve A-small |
| Large input 7 points | Solve A-large |

Problem

You're playing a video game, in which you will get an achievement if you complete all of the levels consecutively without dying. You can play the levels in any order, and each time you play a level you'll either complete it or die. Each level has some probability that you'll complete it, and takes some amount of time. In what order should you play the levels so that the expected time it takes you to get the achievement is minimized? Assume that it takes equally long to beat a level or to die in it, and that you will start again from the first level in your ordering as soon as you die.

Note: If you fail to complete a level, you do not personally die—only your character in the game dies. If that were not the case, only a few people would try to earn this achievement.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow, each of which consists of three lines. The first line of each test case contains a single integer **N**, the number of levels. The second line contains **N** space-separated integers L_i. L_i is the number of seconds level i lasts, which is independent of whether you complete the level or die. The third line contains **N** space-separated integers P_i. P_i is the percent chance that you will **_die_** in any given attempt to complete level i.

Output

For each test case, output one line containing "Case #x: ", where x is the case number (starting from 1), followed by **N** space-separated integers. The jth integer in the list should be the index of the jth level you should attempt to beat in order to minimize the amount of time you expect to spend earning the achievement.

Indices go from 0 to N-1. If there are multiple orderings that would give the same expected time, output the lexicographically least ordering. Out of two orderings, the lexicographically smaller one is the one with the smaller index at the first location where they differ; out of many orderings, the lexicographically least one is the one that is lexicographically smaller than every other ordering.

Limits

1 ≤ **T** ≤ 100.
0 ≤ P_i < 100.

Small dataset

1 ≤ **N** ≤ 20.
L_i = 1.

Large dataset

1 ≤ **N** ≤ 1000.
1 ≤ L_i ≤ 100.

Sample

```
Input            Output

3                Case #1: 0 2 3 1
4                Case #2: 1 0 2
1 1 1 1          Case #3: 2 0 1
50 0 20 20
3
100 10 1
0 50 0
3
100 80 50
40 20 80
```

Note that the second and third samples do not satisfy the constraints for the small input.