BMF (Bi-Modular Friends) are trying to build robots. They divided themselves into smaller groups to build several parts of robots. Two of the groups are assigned to make hands. Robot hands are of two types: left and right. There are N numbers of hands of each type. Each of left hands has unique id between 1 to N and so as for each right hands. For each hand (of both type), a performance factor is measured.

To build a robot it is required to have two hands of different types. The two teams who made hands collaborated to keep the size and shape of the hands synchronized. But they could not synchronize the performance factors. So the experimented with the hands and came up with an idea. To choose each pair of hands they decided to follow following rules:

- Difference between the factors of two hands has to be minimized.
- (If there are multiple candidate pairs) Summation of the factors has to be maximized.
- (If still there are multiple candidate pairs) Pair with left hand of lower id has to be chosen.
- (If still there are multiple candidate pairs) Pair with right hand of lower id has to be chosen.

You have to find the ordered list of all pairs.

Input

Input starts with a line containing an integer T ($T \leq 15$), the number of test cases.

First line of each test case contains N ($1 \leq N \leq 100000$), total number of hands. Second line of input contains N positive integers, performance factors of left hands according to their ids. Again the third line contains N positive integers, performance factors of right hands according to their id. No factor value will be greater than 10^6. Sum of all N among all test cases will be less than 10^6. Counting of ids starts from 1.

Output

In first line, print line '`Case #`x`:`' where x is the test case no, starting from 1.

In next few lines, print the ids of the pairs, formatted as 'l r', where l is the id of left hand and r is the id of right hand.

The list of the pairs has to be exactly the order at which the group has chosen.

Sample Input

```
1
6
1 2 3 4 5 8
4 6 7 8 9 10
```

Sample Output

```
Case #1:
6 4
4 1
5 2
3 3
2 5
1 6
```