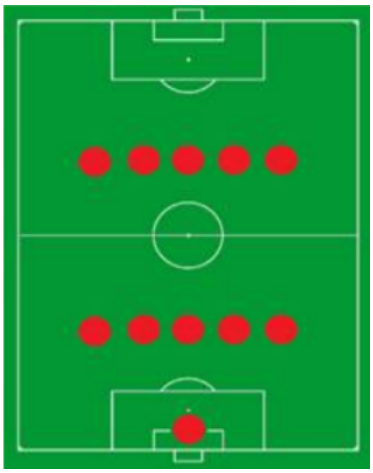
CP #49: Argentina

The Argentine football team coach, the great Diego Maradona, is going to try out a new formation this year. Formation describes how the players are positioned on the pitch. Instead of the conventional 4-4-2 or 4-3-3, he has opted for 5-5. This means there are 5 attackers and 5 defenders.

You have been hired by Argentina Football Federation (AFF) to write a code that will help them figure out which players should take the attacking/defensive positions.



Maradona has given you a list containing the names of the 10 players who will take the field. The attacking ability and the defensive ability of each player are also given. Your job is to figure out which 5 players should take the attacking positions and which 5 should take the defensive positions.

The rules that need to be followed to make the decision are:

- The sum of the attacking abilities of the 5 attackers needs to be maximized
- If there is more than one combination, maximize the sum of the defending abilities of the 5 defenders
- If there is still more than one combination, pick the attackers that come lexicographically earliest.

Input Format

The first line of input contains an integer T (T < 50) that indicates the number of test cases. Each case contains exactly 10 lines. The i-th line contains the name of the i-th player followed by the attacking and defending ability of that player respectively. The length of a players name is at most 20 and consists of lowercase letters only. The attacking/defending abilities are integers in the range [0, 99].

Output Format

The output of each case contains three lines. The first line is the case number starting from 1. The next line contains the name of the 5 attackers in the format '$(A_1, A_2, A_3, A_4, A_5)$' where A_i is the name of an attacker. The next line contains the name of the 5 defenders in the same format. The attackers and defenders names should be printed in lexicographically ascending order. Look at the sample for more details.

Sample Input 0

```
1
sameezahur 20 21
```

```
sohelh 18 9
jaan 17 86
sidky 16 36
shamim 16 18
shadowcoder 12 9
muntasir 13 4
brokenarrow 16 16
emotionalblind 16 12
tanaeem 20 97
```

Sample Output 0

```
Case 1:
(emotionalblind, jaan, sameezahur, sohelh, tanaeem)
(brokenarrow, muntasir, shadowcoder, shamim, sidky)
```


Current Buffer (saved locally, editable) Java 8

```java
import java.io.*;
import java.util.*;
class Player implements Comparable<Player>{
    String name;
    int attack;
    int defend;

    Player(String n, int a,int d){
        this.name = n;
        this.attack = a;
        this.defend = d;
    }

    public int compareTo(Player that){
        if (this.attack>that.attack){
            return 1;
        }
        else if (this.attack<that.attack){
            return -1;
        }
        else{
            if (this.defend>that.defend){
                return -1;
            }
            else if (this.defend<that.defend){
                return 1;
            }
            else{
                return 0;
            }
        }
    }
}

public class Solution {

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. */
        Scanner sc= new Scanner(System.in);
        while (sc.hasNextLine()){
```