Selling Computer Shop

Description

You are a computer shop's owner and due to some circumstances, you find yourself in a financial crisis and you are in a need to sell what exist in it in order to get the maximum profit from the selling process. You find another computers seller and is ready to buy some machines. However, this seller is committed with a specific total load ***L*** since he has only TWO cars and each car can carry a specific load (i.e. car 1 load is ***L1*** and car 2 load is ***L2***).

The problem is:
Given **N** items and 2 arrays (1-based, i.e. the first element is placed at index 1), an array contains the load of each item and the other contains the price to be sold with for each. Now, to get out of this crisis in a smart way, you must achieve the minimum lost, i.e. the maximum profit. So, you need to:

1. Find the maximum profit that you can get from the selling process (satisfying the two cars' load).
2. Find the indices (1-based) of the selected items to be loaded in CAR 1 and the selected items shall be loaded in CAR 2 and achieve the maximum gain.
    - If there're two or more combinations that have the same max total gain, return **ANY of them.**

All prices and loads you have, are integers.

Complexity

With an algorithm that takes a polynomial time

Input: Already Implemented

The first line of input is an integer T (T <= 200), that indicates the number of test cases. Each case consists of multiple lines: first one always contains three integer represents the number of machines exist in the shop (N), then the load of the first car and the load of the second car. Then, this first line is followed by N lines, where each contains two integers: the load and price of each product separated by space.

Output: Already Implemented

1. The maximum profit can get from the selling process.
2. The indices of the products selected to be loaded in the **first car** and those selected to be loaded in the **second car**.

Function: Implement it!

```
int GetMaximumProfit(int car1_load, int car2_load, int N, int[] loads, int[]
            prices, List<int> car1_items, List<int> car2_items)
```

It takes:

1. The load, car 1 can carry
2. The load, car 2 can carry
3. The overall number of products in the shop
4. The array containing the load of each product (**placed starting from index 1 to N**)
5. The array containing the price of each product (**placed starting from index 1 to N**)
6. EMPTY list for car 1 selected items
7. EMPTY list for car 2 selected items

This function responsible to:

1. **Return** the max profit that you can obtain by selling to other seller what are fit in his 2 cars
   [int: **RETURN BY FUNCTION**]
2. **Fill car1_items** with the indices [1-based] of products selected to be loaded in the first car
3. **Fill car2_items** with the indices [1-based] of products selected to be loaded in the second car
   (if there're multiple combinations that have the same max total gain, return ANY of them)
   - To add in list in C#, use the **add** function:
     - car1_items.add(item);

Template

- C# template

Test Cases

#	Input	Output
1	N = 4, car1_load = 5, car2_load = 2 Loads[1] = 1, prices[1] = 20 Loads[2] = 2, prices[2] = 10 Loads[3] = 3, prices[3] = 15 Loads[4] = 5, prices[4] = 25	Max profit = 45 Car1 items = [2, 3] Car2 items = [1]
2	N = 5, car1_load = 7, car2_load = 10 Loads[1] = 7, prices[1] = 42 Loads[2] = 3, prices[2] = 12 Loads[3] = 4, prices[3] = 40 Loads[4] = 5, prices[4] = 25 Loads[5] = 3, prices[5] = 40	Max profit = 134 Car1 items = [1] Car2 items = [2, 3, 5]

C# Help

If you need any help regarding the syntax of C#, **ask any TA**.

Creating 1D array

```
int [] array = new int [size]
```

Creating 2D array

```
int [,] array = new int [size1, size2]
```

Sorting single array

Sort the given array "items" in ascending order

```
Array.Sort(items);
```

Sorting parallel arrays

Sort the first array "master" and re-order the 2nd array "slave" according to this sorting

```
Array.Sort(master, slave);
```