Ali Baba in the Cave

Description

Ali Baba and his thieves enter a cave that contains a set of *n* **types of items** from which they are to select some number of items to be theft. Each item type has a *weight*, a *profit* and a *number of available instances*. Their objective is to choose the set of items that fits the possible load of their Camels and maximizes the profit. Note the following:

1- The array of items is **1-based,** i.e. the first item is placed at index 1 not index 0

2- Each item should be **taken as a whole** (i.e. they can't take part of it)

3- They can take the same item **one or more time** (according to its number of instances)

REQUIRED: Given ***n* triples** where each triple represents the (**weight, profit, number of instances**) of an **item type** and the **Camels possible load**, find:

1. **Maximum profit** that can be loaded on the Camels by the **OPTIMAL WAY**
2. **BONUS 1:** Find maximum profit in **TOTAL AVERAGE TIME** less than **170 ms**
3. **BONUS 2:** **Retrieve list of the items chosen** to get MAX profit and the **number of instances** taken from each item.

Complexity

Your algorithm should take **polynomial time**

Example

N = 4 Load = 10

Weight	Profit	# instances
2	1	2
4	8	2
3	6	2
4	5	2

Max Profit = 20$, as follows:

1. one instance from item2 (profit 8$)
2. two instances from item3 (profit 6$ × 2 = 12$)

Input: **Already Implemented**

First line contains number of test cases. Each case consists of:

1. A line containing the number of items (N) and max camels load
2. N triples, one per line, consisting of item's weight, profit and number of instances.

Output: Already Implemented

For each input case, there should be a line containing the result value.

BONUS: There are two lines follow the result of the MAX obtained PROFIT, which are: line contains the indices of the items chosen (1-based) and the other line contains the number of instances taken from each item.

Function: Implement it!

```
int AliBaba(int camelsLoad, int itemsCount, int[] weights, int[] profits, int[] instances,
                ref int[] items_taken, ref int[] instances_of_items_taken)
```

INPUT:

1. camelsLoad: max load that can be carried by camels
2. itemsCount: number of items inside the cave
3. weights[]: weight of each item [**ONE-BASED** array]
4. profits[]: profit of each item [**ONE-BASED** array]
5. instances[]: number of instances for each item [**ONE-BASED** array]

OUTPUT:

1. It should return the max total profit that can be carried within the given camels' load.
2. **[USED FOR the 2nd BONUS ONLY]** items_taken[]: array initialized with length of itemsCount. You fill it with the indices (1-BASED) of items selected to get the MAX profit.
3. **[USED FOR the 2nd BONUS ONLY]** instances_of_items_taken[]: array initialized with length of itemsCount. You shall fill it with the number of instances taken from each selected item to get the MAX profit.

Template

- C# template

Test Cases

#	Input	Output

1	3 9
	1 5 2
	2 4 1
	3 3 1
		17

2	4 10
	2 1 2
	4 8 2
	3 6 2
	4 5 2
		20

3	4 9
	1 7 3
	3 5 3
	4 2 3
	1 3 2
		32

4	2 3
	4 2 3
	4 6 3
		0

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 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);
```