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)

Requirements:

Given ***N* items** with the **weight, profit & number of instances** of each, and the **Camels possible load** Implement TWO functions,

1. **Function#1:** return **Maximum profit** that can be loaded on the Camels by the **OPTIMAL WAY**

2. **Function#2:** return **list of the items chosen** to get MAX profit and the **number of instances** taken from each item.

Function:

First Function:

```
 int SolveValue(int camelsLoad, int itemsCount, int[] weights, int[] profits, int[]
                                              instances)
```

Parameters

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]

```
<returns>Max total profit
```

Second Function:

```
 Tuple<int, int>[] ConstructSolution(int camelsLoad, int itemsCount, int[] weights,
                               int[] profits , int[] instances)
<returns>Tuple array of the selected items to get MAX profit (stored in Tuple.Item1)
together with the number of instances taken from each item (stored in Tuple.Item2)
OR NULL if no items can be selected
```

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$)

N = 4 Load = 9

Weight	Profit	# instances
1	7	3
3	5	3
4	2	3
1	3	2

Max Profit = 32$, as follows:

```
Item#   #Instances
==============
1               3
2               1
4               2
```

C# Help

TUPLE:

Creating a two-element tuple of integers

```
Tuple<int, int> t = new Tuple<int, int>(5, 7)
```

Accessing

`t.Item1` ➔ return 1st value (5 in the given example)

`t.Item2` ➔ return 2nd value (7 in the given example)

ARRAYS:

Creating 1D array

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

Creating 2D array

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

Length of 1D array

```
int arrayLength = my1DArray.Length
```

Length of 2D array

```
int array1stDim = my2DArray.GetLength(0)

int array2ndDim = my2DArray.GetLength(1)
```

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);
```