Sum to a Number

Description

You are given a sequence of positive integers $a_1, a_2, \ldots a_n$, and a positive integer ***B***. Your goal is to determine if there's exist any subsequence of $a_1, a_2, \ldots a_n$ that sums up to exactly ***B***?
If found:

1. First function should return true.
2. Second function should return the subsequence that forms B.

Else:

1. First function should return false.
2. Second function should return null.

Input: Already Implemented

The first line of input is an integer T (T < 100,000), that indicates the number of test cases. Each case consists: two integers (number to be checked (B) and number of integers (N)), and the N integers.

Output: Already Implemented

1. First function: return the Boolean value (either true or false).
2. Second function: return the subsequence (if any) or null.

Function: Implement it!

First Function:

```
bool SolveValue(int []items, int N, int B)
```

Second Function:

```
int[] ConstructSolution(int []items, int N, int B)
```

Both takes the array of integers (`items`), number of them (N) and the value to be checked (B). If there's any subsequence that sums exactly to B, return true from 1st function and the subsequence itself from the 2nd function. Else, return false and null.

Test Cases

#	Input	Output of 1st Fn	Output of 2nd Fn
1	B = 4, items = [5, 2, 1, 3, 1]	`true`	`2,1,1`
2	B = 5, items = [1, 1, 1, 1, 0]	`false`	`Null`
3	B = 10, items = [5, 7, 2, 1, 3]	`true`	`7,3 OR 5,2,3 OR 7,2,1`
4	B = 15, items = [1, 2, 3, 4, 15]	`true`	`15`

C# Help

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);
```