# CSE 331/503

## Computer Organization

## Homework 2 Report

We need a recursive function using brute force algorithm to see the all possible possibilities in the increasing sub sequences

```
increasingSubSequnce(int[] array , int index , List<Integer> tmpList)
```

And as each recursive function has a base stop statement in this problem the base case is if the index is greater or equal the array's size. Because there is no more left element to check is there any bigger element than my tmp's last element.

```
if(array.length <= index)        Also we are doing printing in this contidition
```

Then we need to check another thing if the tmp array has more element than current longest array. We need to reasign the longest array and size again.

```
if( maxLength < tmpList.size()){
        list = tmpList; maxLength = tmp.size();      // list is longest array
}
```

Then we are doing the comparison for the next added element if the array's next element is bigger than tmpList last element (bigger number) we can add it as a new member of array.

```
if(tmp.size() == 0 || array[index] > tmp.get(tmp.size()-1) ){
tmp.add(array[index]);
```

As a countinue in this condition we are looking for the next bigger element in the list

```
increasingSubSequnceRec(array, index+1, tmp);
```

To see all possibility we are removing the last added element

```
tmp.remove(tmp.size()-1);
```

End of the recursive we have to go throw the getting array so we need a line recursive will countinue

```
increasingSubSequnceRec(array, index+1, tmp);
```

## Examples:

```
25,10,75,9,56,11
5,43,23,6,46,83,32
13,3,9,24,6,12,53,21,65
3,10,7,9,4,11
8,10,5,9,12,14,21,52
111,109,112,100,101,125,124
```

**1)**

```
candidate sequence : [25 75 ]  size =  2
candidate sequence : [25 56 ]  size =  2
candidate sequence : [25 ]  size =  1
candidate sequence : [10 75 ]  size =  2
candidate sequence : [10 56 ]  size =  2
candidate sequence : [10 11 ]  size =  2
candidate sequence : [10 ]  size =  1
candidate sequence : [75 ]  size =  1
candidate sequence : [9 56 ]  size =  2
candidate sequence : [9 11 ]  size =  2
candidate sequence : [9 ]  size =  1
candidate sequence : [56 ]  size =  1
candidate sequence : [11 ]  size =  1
candidate sequence : [0 ]  size =  0
Longest:  candidate sequence : [25 75 ]  size =  2
```

**2)**

```
candidate sequence : [5 43 46 83 ]  size =  4
candidate sequence : [5 43 46 ]  size =  3
candidate sequence : [5 43 83 ]  size =  3
candidate sequence : [5 43 ]  size =  2
candidate sequence : [5 23 46 83 ]  size =  4
candidate sequence : [5 23 46 ]  size =  3
candidate sequence : [5 23 83 ]  size =  3
candidate sequence : [5 23 32 ]  size =  3
candidate sequence : [5 23 ]  size =  2
candidate sequence : [5 6 46 83 ]  size =  4
candidate sequence : [5 6 46 ]  size =  3
candidate sequence : [5 6 83 ]  size =  3
candidate sequence : [5 6 32 ]  size =  3
candidate sequence : [5 6 ]  size =  2
candidate sequence : [5 46 83 ]  size =  3
candidate sequence : [5 46 ]  size =  2
candidate sequence : [5 83 ]  size =  2
candidate sequence : [5 32 ]  size =  2
candidate sequence : [5 ]  size =  1
candidate sequence : [43 46 83 ]  size =  3
candidate sequence : [43 46 ]  size =  2
candidate sequence : [43 83 ]  size =  2
candidate sequence : [43 ]  size =  1
candidate sequence : [23 46 83 ]  size =  3
candidate sequence : [23 46 ]  size =  2
candidate sequence : [23 83 ]  size =  2
candidate sequence : [23 32 ]  size =  2
candidate sequence : [23 ]  size =  1
candidate sequence : [6 46 83 ]  size =  3
candidate sequence : [6 46 ]  size =  2
candidate sequence : [6 83 ]  size =  2
candidate sequence : [6 32 ]  size =  2
candidate sequence : [6 ]  size =  1
candidate sequence : [46 83 ]  size =  2
candidate sequence : [46 ]  size =  1
candidate sequence : [83 ]  size =  1
candidate sequence : [32 ]  size =  1
candidate sequence : [0 ]  size =  0
Longest:  candidate sequence : [5 43 46 83 ]  size =  4
```

**3)**

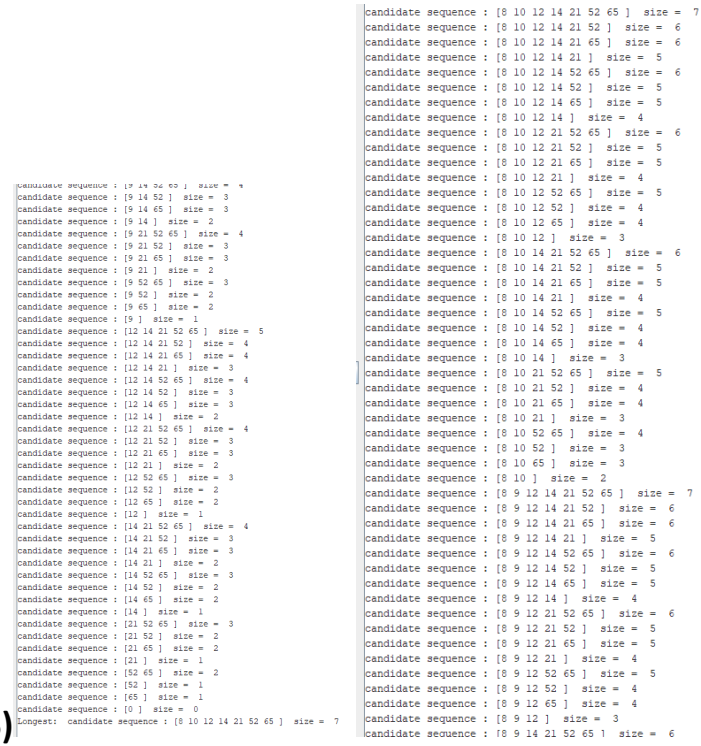
```
candidate sequence : [24 53 65 ]  size =  3
candidate sequence : [24 53 ]  size =  2
candidate sequence : [24 65 ]  size =  2
candidate sequence : [24 ]  size =  1
candidate sequence : [6 12 53 65 ]  size =  4
candidate sequence : [6 12 53 ]  size =  3
candidate sequence : [6 12 21 65 ]  size =  4
candidate sequence : [6 12 21 ]  size =  3
candidate sequence : [6 12 65 ]  size =  3
candidate sequence : [6 12 ]  size =  2
candidate sequence : [6 53 65 ]  size =  3
candidate sequence : [6 53 ]  size =  2
candidate sequence : [6 21 65 ]  size =  3
candidate sequence : [6 21 ]  size =  2
candidate sequence : [6 65 ]  size =  2
candidate sequence : [6 ]  size =  1
candidate sequence : [12 53 65 ]  size =  3
candidate sequence : [12 53 ]  size =  2
candidate sequence : [12 21 65 ]  size =  3
candidate sequence : [12 21 ]  size =  2
candidate sequence : [12 65 ]  size =  2
candidate sequence : [12 ]  size =  1
candidate sequence : [53 65 ]  size =  2
candidate sequence : [53 ]  size =  1
candidate sequence : [21 65 ]  size =  2
candidate sequence : [21 ]  size =  1
candidate sequence : [65 ]  size =  1
candidate sequence : [0 ]  size =  0
Longest:  candidate sequence : [3 9 24 53 65 ]  size =  5
```

```
candidate sequence : [13 24 53 65 ]  size =  4
candidate sequence : [13 24 53 ]  size =  3
candidate sequence : [13 24 65 ]  size =  3
candidate sequence : [13 24 ]  size =  2
candidate sequence : [13 53 65 ]  size =  3
candidate sequence : [13 53 ]  size =  2
candidate sequence : [13 21 65 ]  size =  3
candidate sequence : [13 21 ]  size =  2
candidate sequence : [13 65 ]  size =  2
candidate sequence : [13 ]  size =  1
candidate sequence : [3 9 24 53 65 ]  size =  5
candidate sequence : [3 9 24 53 ]  size =  4
candidate sequence : [3 9 24 65 ]  size =  4
candidate sequence : [3 9 24 ]  size =  3
candidate sequence : [3 9 12 53 65 ]  size =  5
candidate sequence : [3 9 12 53 ]  size =  4
candidate sequence : [3 9 12 21 65 ]  size =  5
candidate sequence : [3 9 12 21 ]  size =  4
candidate sequence : [3 9 12 65 ]  size =  4
candidate sequence : [3 9 12 ]  size =  3
candidate sequence : [3 9 53 65 ]  size =  4
candidate sequence : [3 9 53 ]  size =  3
candidate sequence : [3 9 21 65 ]  size =  4
candidate sequence : [3 9 21 ]  size =  3
candidate sequence : [3 9 65 ]  size =  3
candidate sequence : [3 9 ]  size =  2
candidate sequence : [3 24 53 65 ]  size =  4
candidate sequence : [3 24 53 ]  size =  3
candidate sequence : [3 24 65 ]  size =  3
candidate sequence : [3 24 ]  size =  2
candidate sequence : [3 6 12 53 65 ]  size =  5
candidate sequence : [3 6 12 53 ]  size =  4
candidate sequence : [3 6 12 21 65 ]  size =  5
```

**4)**

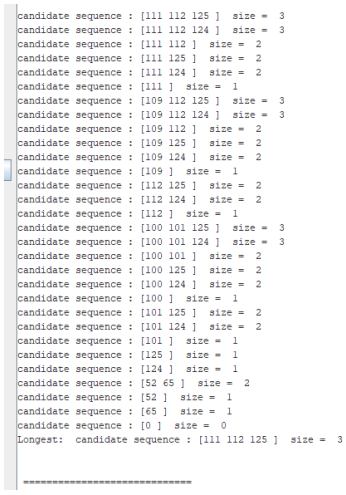
```
candidate sequence : [9 11 53 65 ]  size =  4
candidate sequence : [9 11 53 ]  size =  3
candidate sequence : [9 11 21 65 ]  size =  4
candidate sequence : [9 11 21 ]  size =  3
candidate sequence : [9 11 65 ]  size =  3
candidate sequence : [9 11 ]  size =  2
candidate sequence : [9 53 65 ]  size =  3
candidate sequence : [9 53 ]  size =  2
candidate sequence : [9 21 65 ]  size =  3
candidate sequence : [9 21 ]  size =  2
candidate sequence : [9 65 ]  size =  2
candidate sequence : [9 ]  size =  1
candidate sequence : [4 11 53 65 ]  size =  4
candidate sequence : [4 11 53 ]  size =  3
candidate sequence : [4 11 21 65 ]  size =  4
candidate sequence : [4 11 21 ]  size =  3
candidate sequence : [4 11 65 ]  size =  3
candidate sequence : [4 11 ]  size =  2
candidate sequence : [4 53 65 ]  size =  3
candidate sequence : [4 53 ]  size =  2
candidate sequence : [4 21 65 ]  size =  3
candidate sequence : [4 21 ]  size =  2
candidate sequence : [4 65 ]  size =  2
candidate sequence : [4 ]  size =  1
candidate sequence : [11 53 65 ]  size =  3
candidate sequence : [11 53 ]  size =  2
candidate sequence : [11 21 65 ]  size =  3
candidate sequence : [11 21 ]  size =  2
candidate sequence : [11 65 ]  size =  2
candidate sequence : [11 ]  size =  1
candidate sequence : [53 65 ]  size =  2
candidate sequence : [53 ]  size =  1
candidate sequence : [21 65 ]  size =  2
candidate sequence : [21 ]  size =  1
candidate sequence : [65 ]  size =  1
candidate sequence : [0 ]  size =  0
Longest:  candidate sequence : [3 7 9 11 53 65 ]  size =  6
```

```
candidate sequence : [3 10 11 53 65 ]  size =  5
candidate sequence : [3 10 11 53 ]  size =  4
candidate sequence : [3 10 11 21 65 ]  size =  5
candidate sequence : [3 10 11 21 ]  size =  4
candidate sequence : [3 10 11 65 ]  size =  4
candidate sequence : [3 10 11 ]  size =  3
candidate sequence : [3 10 53 65 ]  size =  4
candidate sequence : [3 10 53 ]  size =  3
candidate sequence : [3 10 21 65 ]  size =  4
candidate sequence : [3 10 21 ]  size =  3
candidate sequence : [3 10 65 ]  size =  3
candidate sequence : [3 10 ]  size =  2
candidate sequence : [3 7 9 11 53 65 ]  size =  6
candidate sequence : [3 7 9 11 53 ]  size =  5
candidate sequence : [3 7 9 11 21 65 ]  size =  6
candidate sequence : [3 7 9 11 21 ]  size =  5
candidate sequence : [3 7 9 11 65 ]  size =  5
candidate sequence : [3 7 9 11 ]  size =  4
candidate sequence : [3 7 9 53 65 ]  size =  5
candidate sequence : [3 7 9 53 ]  size =  4
candidate sequence : [3 7 9 21 65 ]  size =  5
candidate sequence : [3 7 9 21 ]  size =  4
candidate sequence : [3 7 9 65 ]  size =  4
candidate sequence : [3 7 9 ]  size =  3
candidate sequence : [3 7 11 53 65 ]  size =  5
candidate sequence : [3 7 11 53 ]  size =  4
candidate sequence : [3 7 11 21 65 ]  size =  5
candidate sequence : [3 7 11 21 ]  size =  4
candidate sequence : [3 7 11 65 ]  size =  4
candidate sequence : [3 7 11 ]  size =  3
candidate sequence : [3 7 53 65 ]  size =  4
candidate sequence : [3 7 53 ]  size =  3
candidate sequence : [3 7 21 65 ]  size =  4
candidate sequence : [3 7 21 ]  size =  3
candidate sequence : [3 7 65 ]  size =  3
candidate sequence : [3 7 ]  size =  2
candidate sequence : [3 9 11 53 65 ]  size =  5
candidate sequence : [3 9 11 53 ]  size =  4
candidate sequence : [3 9 11 21 65 ]  size =  5
candidate sequence : [3 9 11 21 ]  size =  4
candidate sequence : [3 9 11 65 ]  size =  4
candidate sequence : [3 9 11 ]  size =  3
```




5)




6)

Results are seeing in the mars. I try to put all result but some test scenario has too many result so I had to crop the result but you can look at the all result in .asm code execute on the mars simulator.
0 is not the getting valid number [0] represent the end of the array

When I doing the code I try to code most functional way i didnt want to data store data on register. Register just making the operation like assignArray, LenghtOfArray, RemoveLastElement,AddNewElement, GetNthAddress … this functions just taking the $a1 start address of array and return it with $v1 or $v0 any other register just using operation not storing data.

`array[index]> tmp.get(tmp.size()-1)` using this comparision not looking the all variation the eliminate. Just countinue on valid arrays.

As a missing part writing file. But. I know how to do it. Reverse of the reading. Convert integer value to byte using itoa function then write the file.

**Muhammed Bedir ULUCAY**

**1901042697**