## Code Explanation

```
void
subsetOfArray(int* currSubset, int currSubInd, int* inputArr, int index, int* longestSeq, int* maxLen){

    if(index >= inputArr[0]+1 ) {

        if(*maxLen < currSubInd){
            copyArray(longestSeq, currSubset, currSubInd);
            *maxLen = currSubInd;
        }
        printArray(currSubset, currSubInd);
        return;
    }

    if(currSubInd == 0 || currSubset[currSubInd-1] <= inputArr[index]){
        currSubset[currSubInd++] = inputArr[index];
        subsetOfArray(currSubset, currSubInd, inputArr, index+1, longestSeq, maxLen);
        currSubInd--;
    }
    subsetOfArray(currSubset, currSubInd, inputArr, index+1, longestSeq, maxLen);
}
```

Above recursive function finds all increasing subsequences of given array.
**Base Case:**

- if obtained subsequent length is greater than previous one, it is accepted as new longest.
- Each increasing sequence is printed on screen

**Recursive Case:**

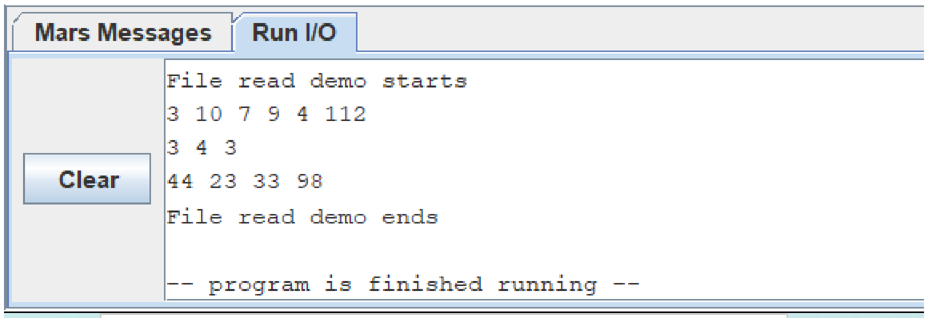
- Each recursion produce 2 more recursion
  - first call includes current item of array
  - second call just increment array index by one (does not take current item)

The whole C implementation is also exist with in the homework folder.

## Missing Parts

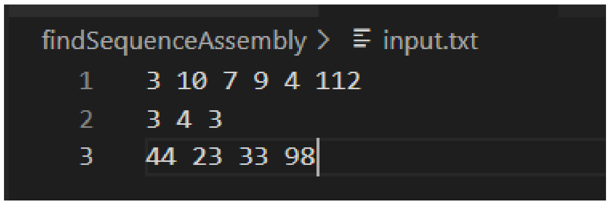
File reading and writing part is not finished, but I could read input file and write its content on terminal of Mars.(At the beginning of execution)

## Test Cases

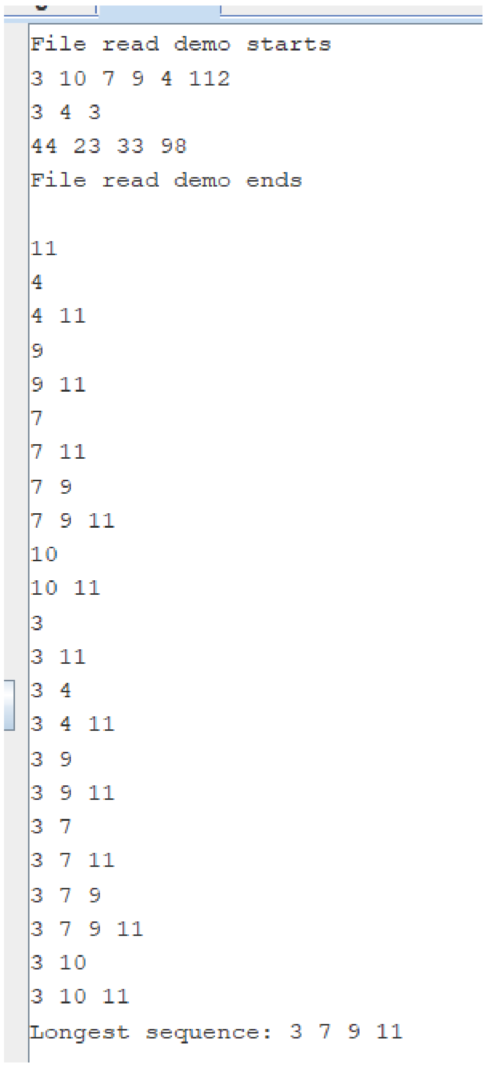


```
File read demo starts
3 10 7 9 4 112
3 4 3
44 23 33 98
File read demo ends

-- program is finished running --
```

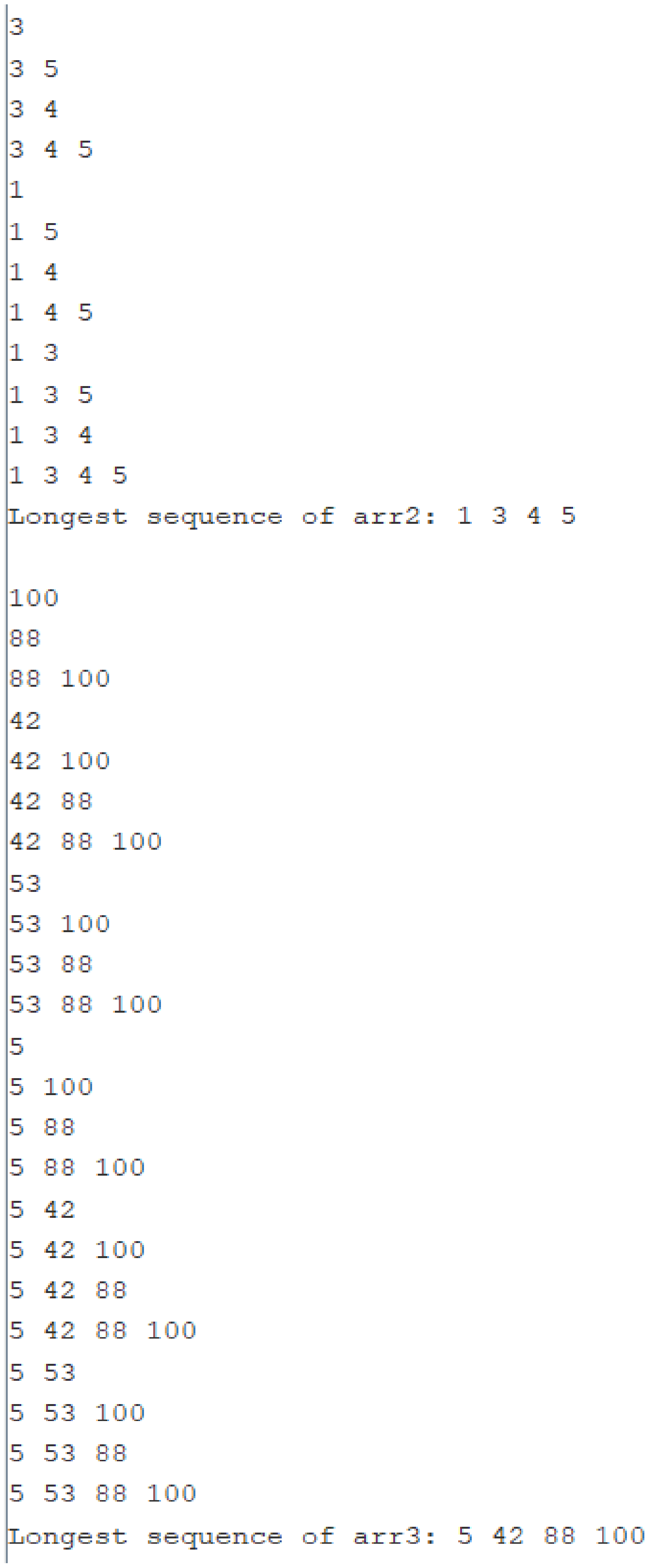


```
3 10 7 9 4 112
3 4 3
44 23 33 98
```



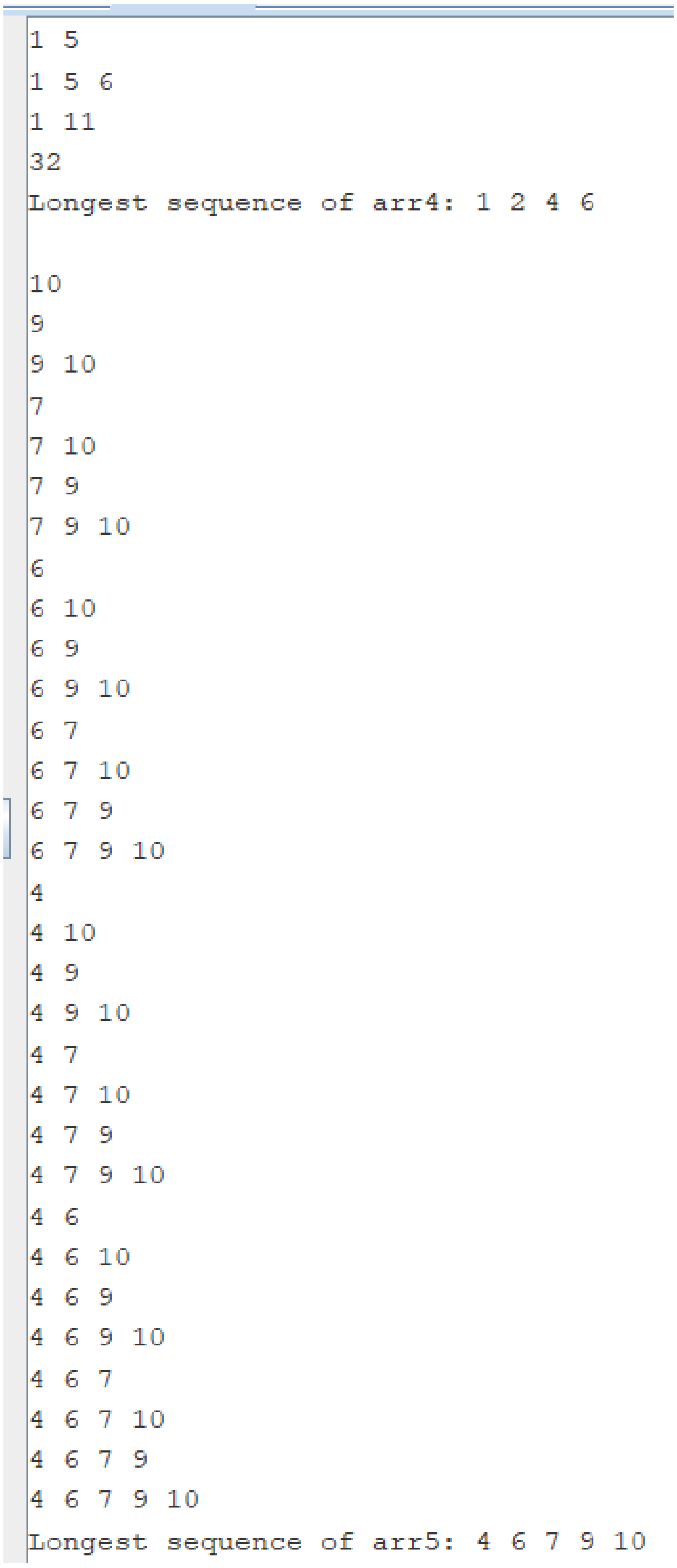
```
File read demo starts
3 10 7 9 4 112
3 4 3
44 23 33 98
File read demo ends

11
4
4 11
9
9 11
7
7 11
7 9
7 9 11
10
10 11
3
3 11
3 4
3 4 11
3 9
3 9 11
3 7
3 7 11
3 7 9
3 7 9 11
3 10
3 10 11
Longest sequence: 3 7 9 11
```



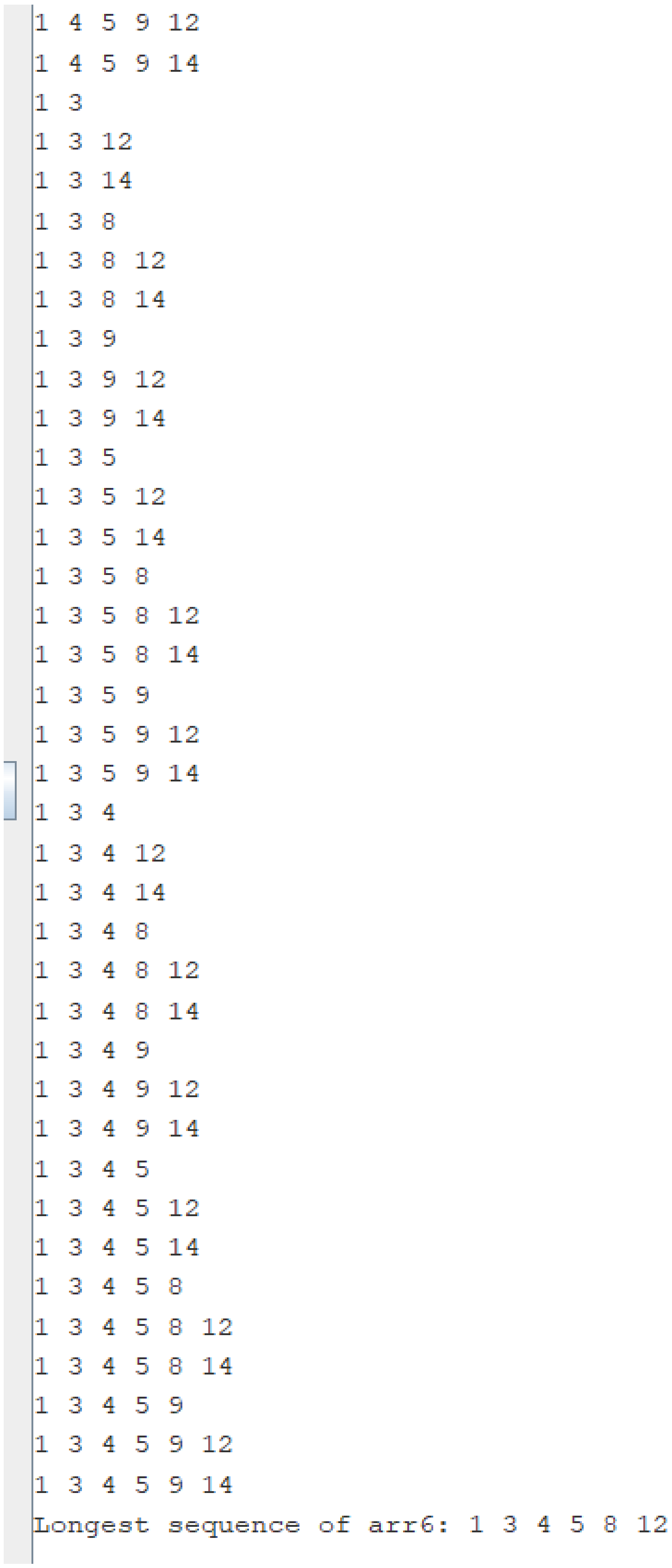
```
3
3 5
3 4
3 4 5
1
1 5
1 4
1 4 5
1 3
1 3 5
1 3 4
1 3 4 5
Longest sequence of arr2: 1 3 4 5

100
88
88 100
42
42 100
42 88
42 88 100
53
53 100
53 88
53 88 100
5
5 100
5 88
5 88 100
5 42
5 42 100
5 42 88
5 42 88 100
5 53
5 53 100
5 53 88
5 53 88 100
Longest sequence of arr3: 5 42 88 100
```



```
1 5
1 5 6
1 11
32
Longest sequence of arr4: 1 2 4 6

10
9
9 10
7
7 10
7 9
7 9 10
6
6 10
6 9
6 9 10
6 7
6 7 10
6 7 9
6 7 9 10
4
4 10
4 9
4 9 10
4 7
4 7 10
4 7 9
4 7 9 10
4 6
4 6 10
4 6 9
4 6 9 10
4 6 7
4 6 7 10
4 6 7 9
4 6 7 9 10
Longest sequence of arr5: 4 6 7 9 10
```



```
1 4 5 9 12
1 4 5 9 14
1 3
1 3 12
1 3 14
1 3 8
1 3 8 12
1 3 8 14
1 3 9
1 3 9 12
1 3 9 14
1 3 5
1 3 5 12
1 3 5 14
1 3 5 8
1 3 5 8 12
1 3 5 8 14
1 3 5 9
1 3 5 9 12
1 3 5 9 14
1 3 4
1 3 4 12
1 3 4 14
1 3 4 8
1 3 4 8 12
1 3 4 8 14
1 3 4 9
1 3 4 9 12
1 3 4 9 14
1 3 4 5
1 3 4 5 12
1 3 4 5 14
1 3 4 5 8
1 3 4 5 8 12
1 3 4 5 8 14
1 3 4 5 9
1 3 4 5 9 12
1 3 4 5 9 14
Longest sequence of arr6: 1 3 4 5 8 12
```