

FUNCTIONS AND DESCRIPTION:

resetElements: resets all .data block

clear: function that is used in foo. Clears arrays

printNewLine: just prints new line

write: writes the candidate solutions to the console

write2File: writes the **longestSubSeq** array that includes solution to the text file.

ltoa: ltoa function. Converts Int to string. Because when we write to the file we need it to be string.

Handwritten diagram illustrating the conversion of the number 123 to its string representation using modulo and division operations:

$$\begin{array}{l} 123 \text{ mod } 10 \rightarrow 3 \xRightarrow{\text{int}} 3 - '0' = '3' \xRightarrow{\text{char}} \text{add buffer} \\ \downarrow \\ 123/10 \rightarrow 12 \rightarrow \text{mod } 10 \rightarrow 2 \Rightarrow 2 - '0' = '2' \rightarrow \text{add buffer} \\ \downarrow \\ 1 \rightarrow \text{mod } 10 \rightarrow 1 \Rightarrow 1 - '0' = '1' \rightarrow \text{add buffer} \end{array}$$

foo (longest increasing Subsequence algorithm): This is the Procedure that does the actual work. Looks at the array and find longest increasing sequence and candidate solutions.

Pseudo code of algorithm:

At the first, first index is taken: $\text{arr} = [3]$

after that the second for loop (inner of for1) begins from **i+1th** index: $k=i+1$

If($3 < 10$) $\text{arr} = [3, 10]$

After that third loop (inner of for2) begins from **k+1th** index: $m = k+1$

If($10 < 7$) $[3, 10, 7]$

Look at the other indexes in for3

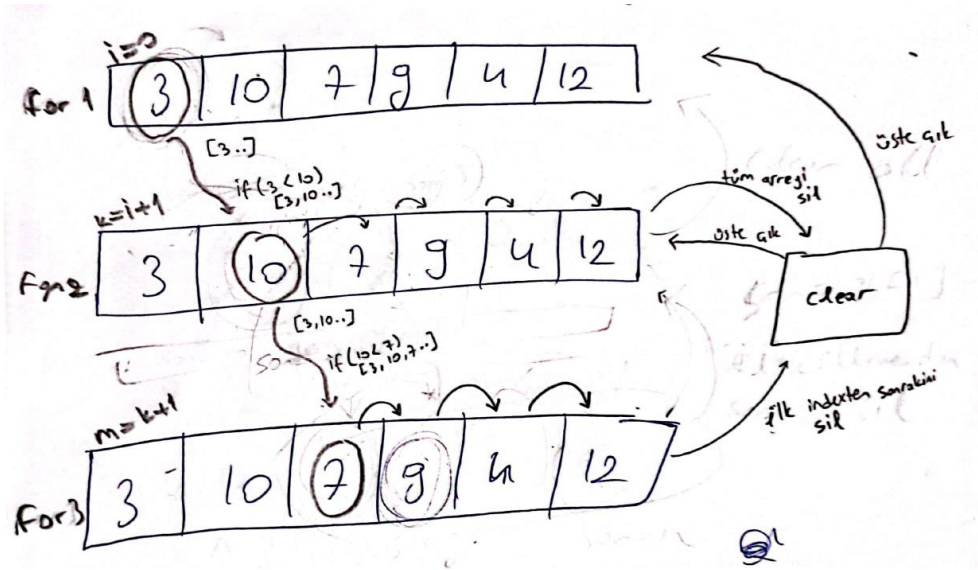
After $m = \text{size}$, go clear function (clear function clears all indexes other than first when loop3 to loop2)

After cleaning, go upper for (for2) and $k++$.

Enter the for3 for all for2 loop indexes.

After $k = \text{size}$ go clear function (clear function clears all indexes when loop2 to loop1)

Go for1. And carry out this process for other i indexes.

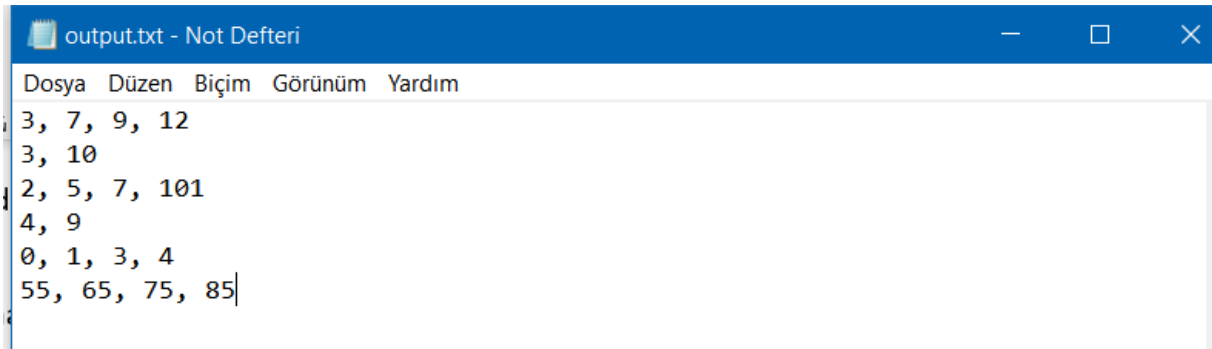


TEST CASES AND OUTPUTS:

```
#test arrays:
givenArr1: .word 3,10,7,9,4,12
givenArr2: .word 3,10,7
givenArr3: .word 10,9,2,5,3,7,101,18
givenArr4: .word 12,4,9,7,10,3
givenArr5: .word 0,1,0,3,3,3,4
givenArr6: .word 55,65,75,85
```

Test cases called one by one in main and the size of the arrays must multiplied by 4 (because of that .word is 4 byte) and entered by hand.

The output is printed out a txt file that name is "output.txt"



All Candidate Solutions are printed to the console screen like that:

```
3 - 3 10 - 3 10 12 - 3 7 - 3 7 9 - 3 7 9 12 - 3 9 - 3 9 12 - 3 4 - 3 4 12 - 3 12 - 10 - 10 12 - 7 - 7 9 - 7 9 12 - 7 12 - 9 - 9 12 - 4 - 4 12 - 12 -  
3 - 3 10 - 3 7 - 10 - 7 -  
10 - 10 101 - 9 - 9 101 - 2 - 2 5 - 2 5 7 - 2 5 7 101 - 2 3 - 2 3 7 - 2 3 7 101 - 2 7 - 2 7 101 - 2 101 - 5 - 5 7 - 5 7 101 - 5 101 - 3 - 3 7 - 3 7 101 - 3 101 - 7 - 7 101 - 101 -  
12 - 4 - 4 9 - 4 7 - 9 - 7 - 3 -  
0 - 0 1 - 0 1 3 - 0 1 3 4 - 0 3 - 0 3 4 - 0 3 - 0 3 4 - 0 3 - 0 3 4 - 0 4 - 1 - 1 3 - 1 3 4 - 1 3 - 1 3 4 - 1 3 - 1 3 4 - 1 4 - 0 - 0 3 - 0 3 4 - 0 3 - 0 3 4 - 0 3 - 0 3 4 - 0 4 - 3 - 3 4 - 3 - 3 4 - 3 - 3 4  
55 - 55 65 - 55 65 75 - 55 65 75 85 - 55 75 - 55 75 85 - 55 85 - 65 - 65 75 - 65 75 85 - 65 85 - 75 - 75 85 - 85 -  
... ..
```

TIME AND SPACE COMPLEXITY:

Time complexity of the foo function that finds the longest increasing sub sequence is **$O(n^3)$** because there are 3 for loop that are nested. Inside and outside of the for loops takes constant time. So

Time complexity of the foo function that finds the longest increasing sub sequence is **$O(n)$** . There are 2 n-element arrays in that function.

Missing parts:

size of longest subsequence is not written down the file

Reading file feature is not done.

Hacı Hasan Savan - 1901042704