FUNCTIONS AND DESCRIPTION:

resetElements: resets all .data block

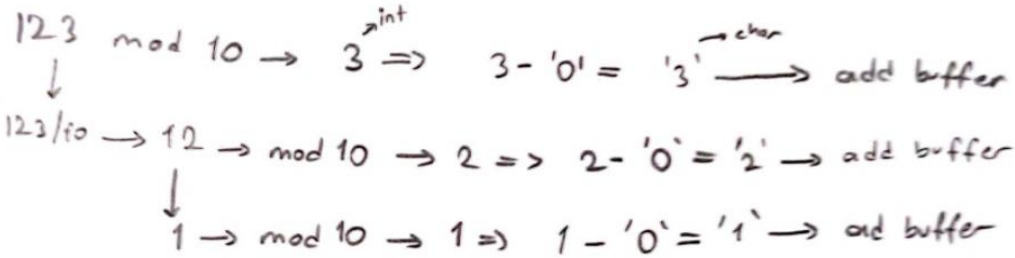
clear: function that is used in foo. Clears arrays

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.

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



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: arr = [3]

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

If(3<10) 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

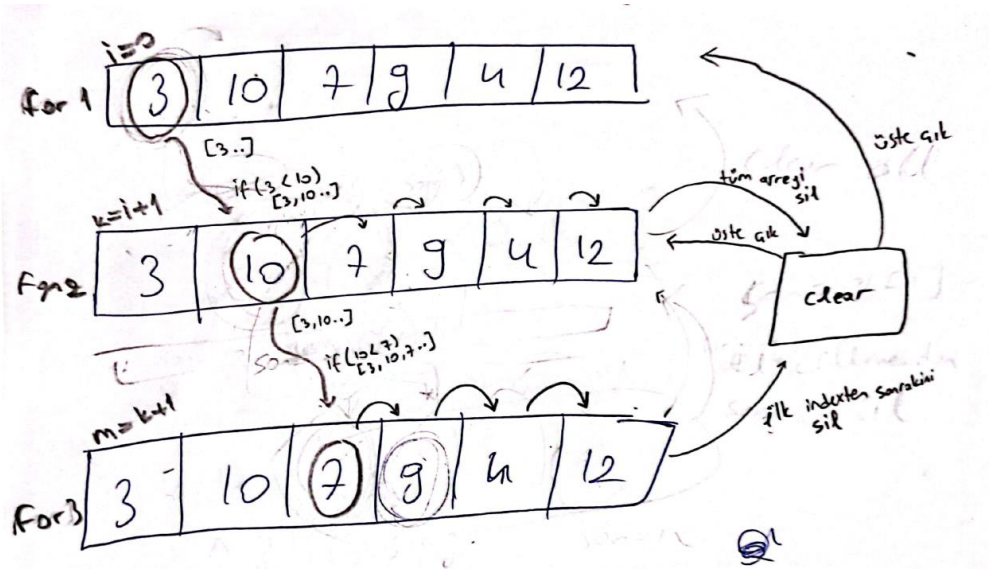
After m=size, go clear function (clear function clears all indexes other than first when loop3 to loop2)

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

Enter the for3 for all for2 loop indexes.

After k=size go clear function (clear function clears all indexes when loop2 to loop1)

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

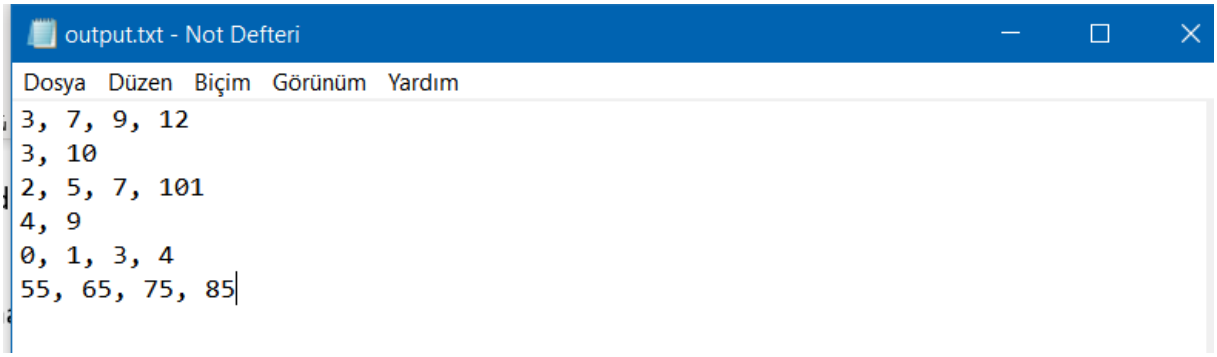


TEST CASES AND OUTPUTS:

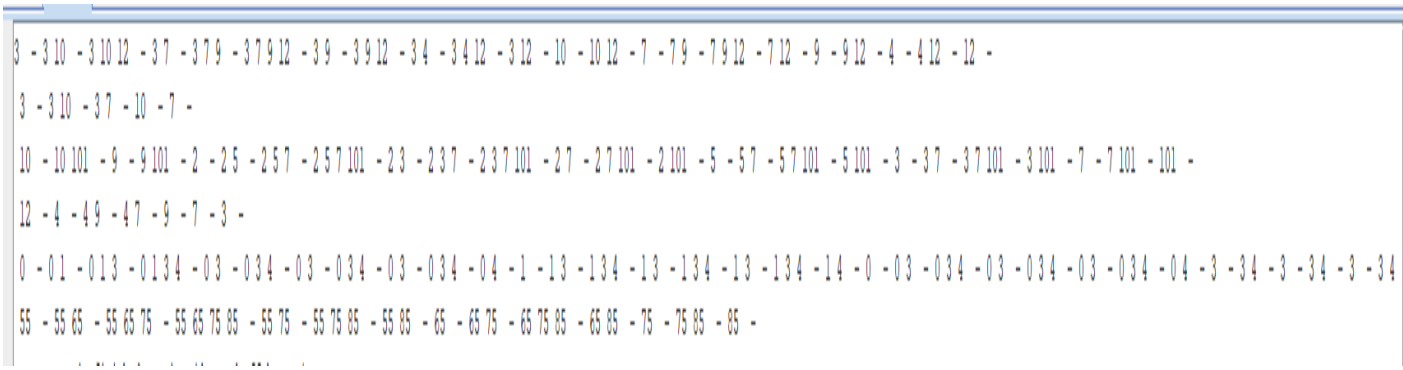
```
#test arrays:
givenArr1: .word 3,10,7,9,4,12
givenArr2: .word 3,10,7
givenArr3: .word 10,9,2,5,3,7,101,18
givenArr4: .word 12,4,9,7,10,3
givenArr5: .word 0,1,0,3,3,3,4
givenArr6: .word 55,65,75,85
```

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:



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