PSEUDOCODE:

findLongest: → recursive procedure

time complexity of findLongest: $\Theta(N^2 \log N)$

space complexity of findLongest: $O(N^2)$

```
findLongest:
    Θ(1) if array[index] equal to array[length] {
            0 assign to $t3
    Θ(1)    if temp[index] equal to 0 {
            } exit(0);
            else{
            } call the printArray procedure
         }
```

→ for subsequence order I create the temp Array

→ I create the general array.

```
    Θ(n) if array.length > temp.length && maxSize >= array[index]
    Θ(n)    for i to maxSize
                if array[index] > temp[index]
                   i = temp[index];
                }
            }
         call check procedure
```

→ time complexity: $\Theta(n)$ space complexity: $O(n)$

→ procedure

```
check:{
        if temp[index] < array[index] {
           array(index) = temp[index]
           array[index++];
}
```

→ I control the increasing subsequence in this procedure → time complexity: $\Theta(1)$ space complexity: $O(1)$

```
call findLongest procedure (recursion)
   temp[index++];
call findLongest;
temp[index++], array[index++];
```

→ time complexity: $O(N \log N)$ space '' '' : $O(N)$

→ I proceed in temp[index] and I go back findLongest procedure

```
printArray:{
    for i to maxSize
        print temp[index]
        print comma
        printLine
        if array[index] != temp[index]
        call printArray
```

→ I print the temp array becouse I keep the subsequence numbers in this array.

→ time complexity: $\Theta(n)$ space complexity: $O(1)$

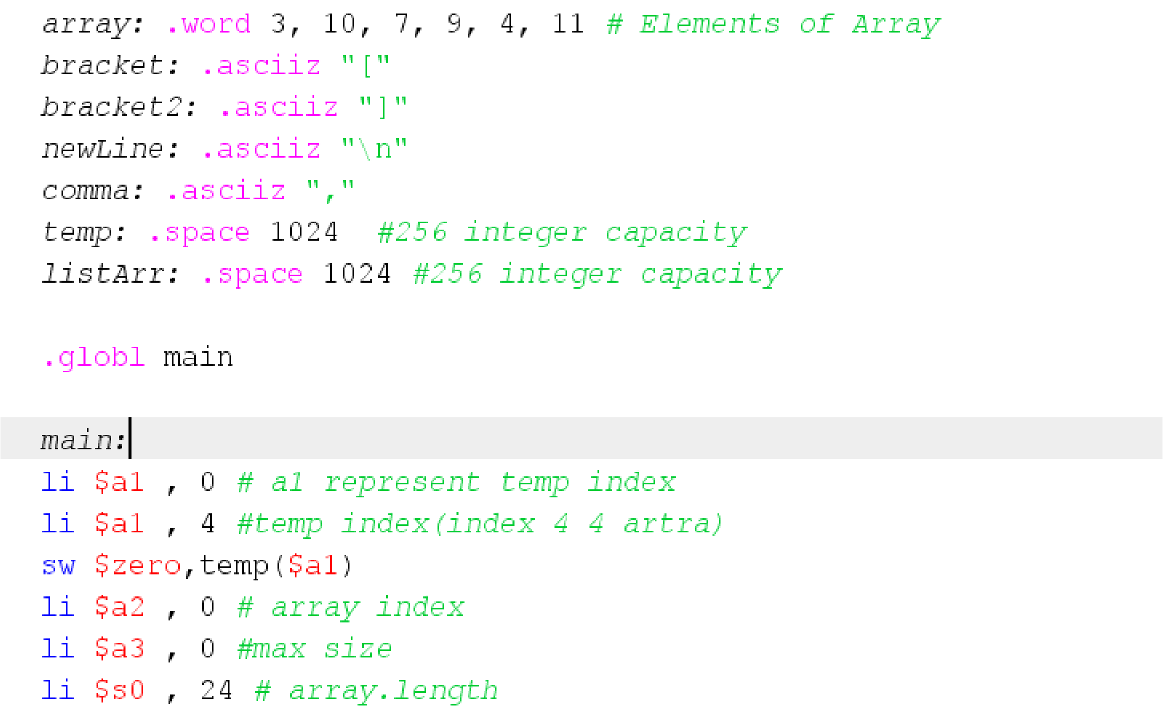
GEBZE TECHNICAL UNIVERSITY

CSE 331/503

HMW2 REPORT:

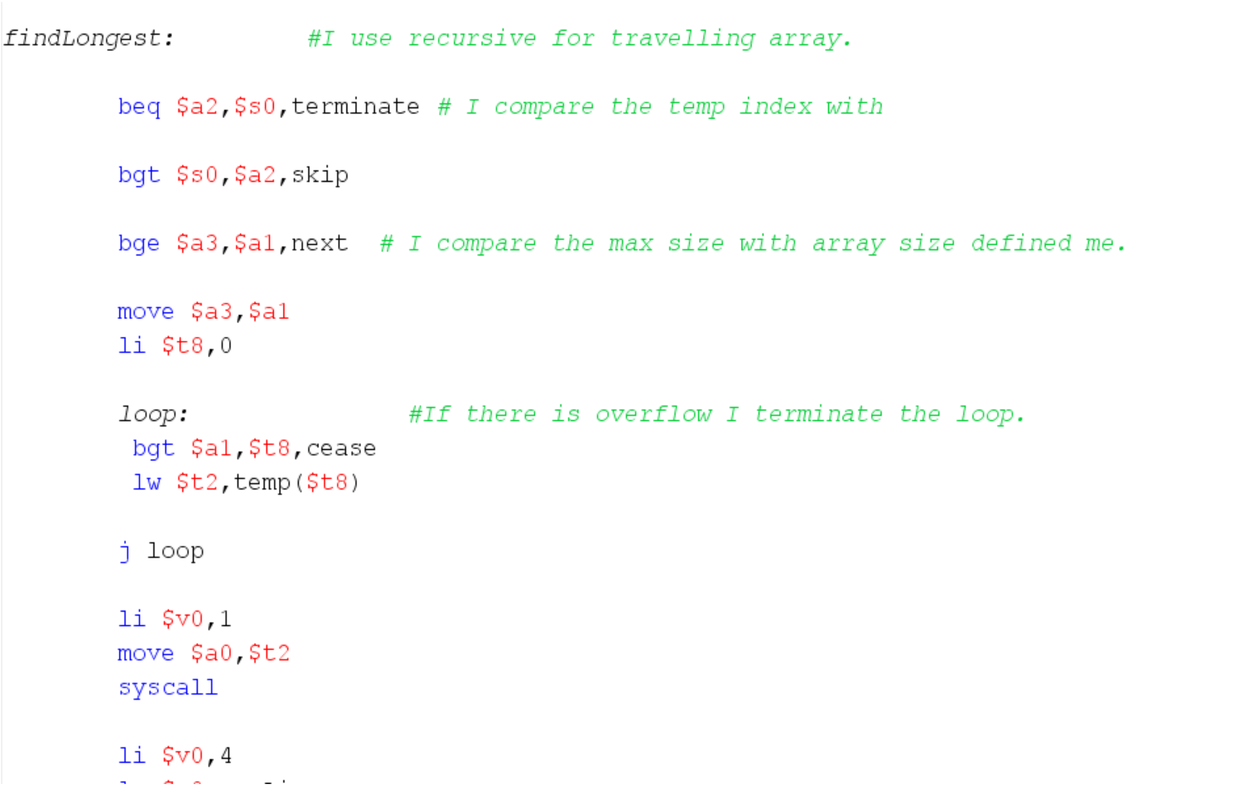
Serdil Anıl Ünlü

1801042672

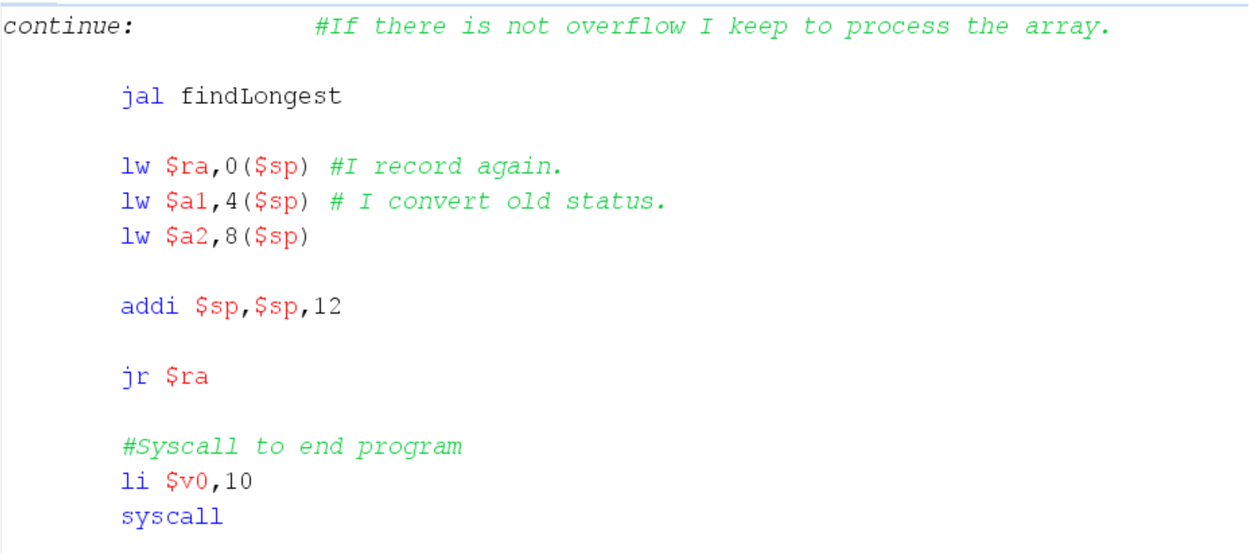


```
array: .word 3, 10, 7, 9, 4, 11 # Elements of Array
bracket: .asciiz "["
bracket2: .asciiz "]"
newLine: .asciiz "\n"
comma: .asciiz ","
temp: .space 1024  #256 integer capacity
listArr: .space 1024 #256 integer capacity

.globl main

main:
li $a1 , 0 # a1 represent temp index
li $a1 , 4 #temp index(index 4 4 artra)
sw $zero,temp($a1)
li $a2 , 0 # array index
li $a3 , 0 #max size
li $s0 , 24 # array.length
```

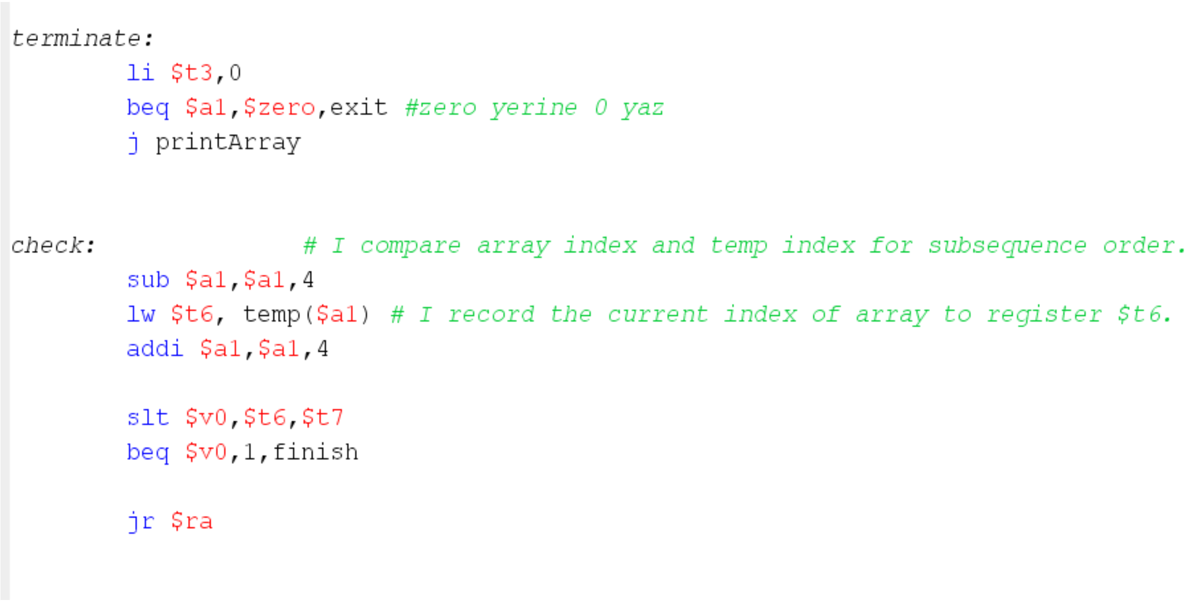
```
jal findLongest # Call the recursive function

la $s7,array            #I assign to array address and I print the 6 different array like this.
li $t0,28    # I record the numbers of array with this way.
li $t1,35
li $t2,3
li $t3,102
li $t4,50
sw $t0,0($s7)
sw $t1,4($s7)
sw $t2,8($s7)
sw $t3,12($s7)
sw $t4,16($s7)
li $a1 , 0 #temp index
li $a1 , 4 #temp index, I increase 4 by 4
sw $zero,temp($a1)
li $a2 , 0 # array index
li $a3 , 0 #max size
li $s0 , 20 # array length


jal findLongest # Call the recursive function
```



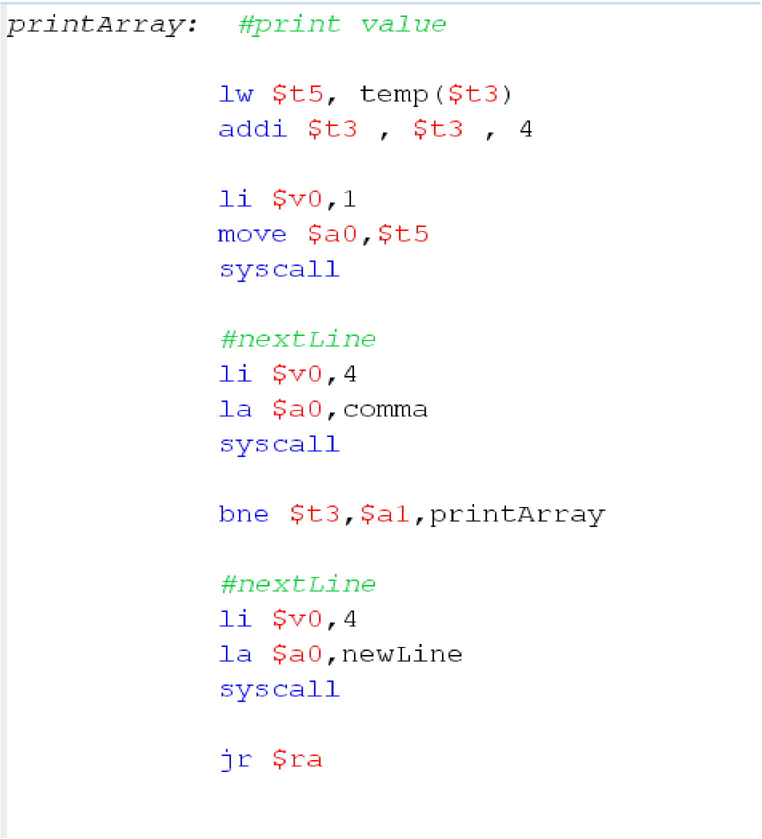
```
findLongest:          #I use recursive for travelling array.

        beq $a2,$s0,terminate # I compare the temp index with

        bgt $s0,$a2,skip

        bge $a3,$a1,next  # I compare the max size with array size defined me.

        move $a3,$a1
        li $t8,0

        loop:                #If there is overflow I terminate the loop.
         bgt $a1,$t8,cease
         lw $t2,temp($t8)

        j loop

        li $v0,1
        move $a0,$t2
        syscall

        li $v0,4
```



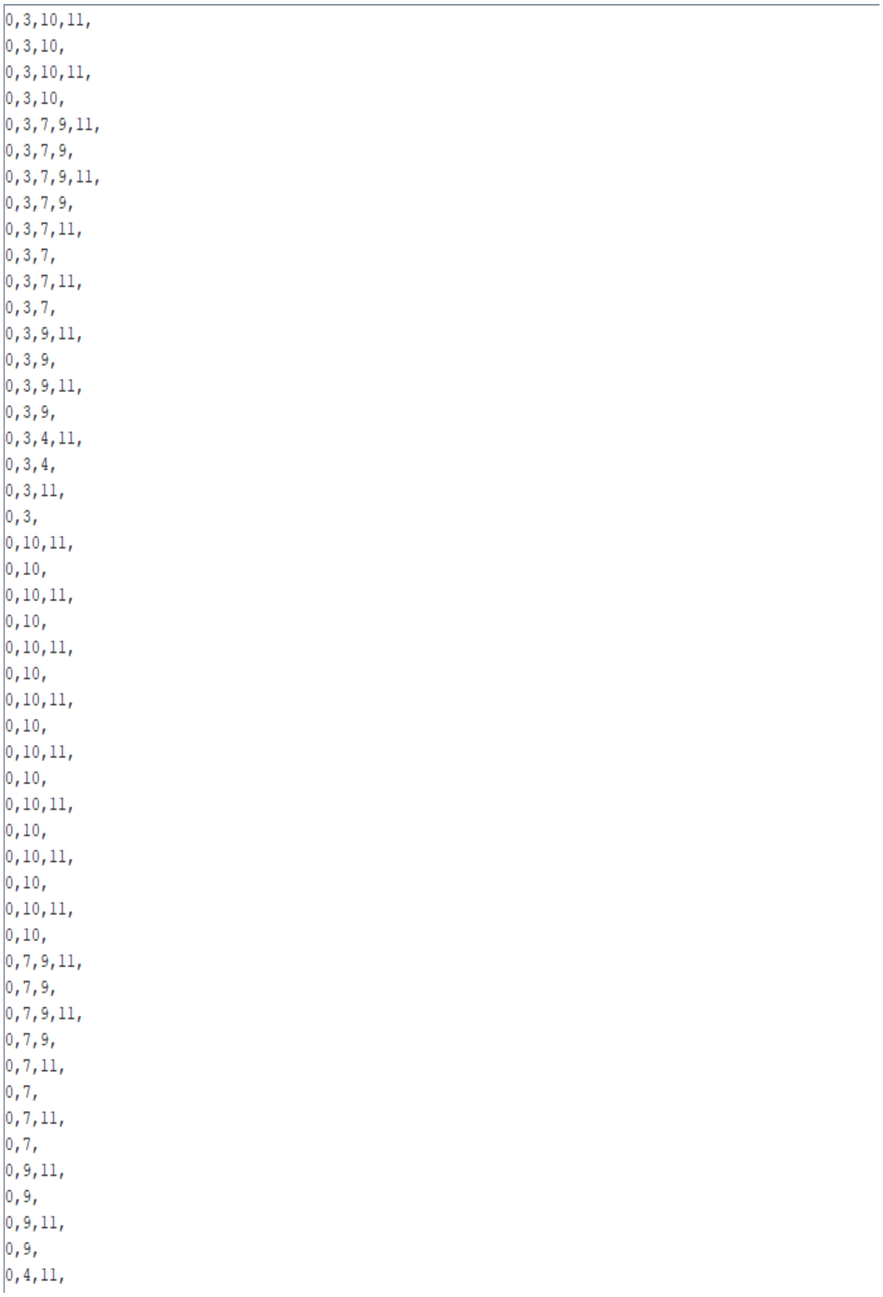
```
continue:                #If there is not overflow I keep to process the array.

        jal findLongest

        lw $ra,0($sp) #I record again.
        lw $a1,4($sp) # I convert old status.
        lw $a2,8($sp)

        addi $sp,$sp,12

        jr $ra

        #Syscall to end program
        li $v0,10
        syscall
```



```
terminate:
        li $t3,0
        beq $a1,$zero,exit #zero yerine 0 yaz
        j printArray


check:              # I compare array index and temp index for subsequence order.
        sub $a1,$a1,4
        lw $t6, temp($a1) # I record the current index of array to register $t6.
        addi $a1,$a1,4

        slt $v0,$t6,$t7
        beq $v0,1,finish

        jr $ra
```

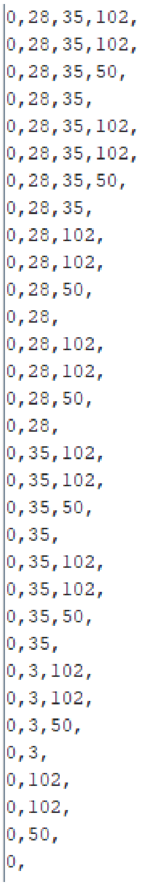


```
printArray:  #print value

            lw $t5, temp($t3)
            addi $t3 , $t3 , 4

            li $v0,1
            move $a0,$t5
            syscall

            #nextLine
            li $v0,4
            la $a0,comma
            syscall

            bne $t3,$a1,printArray

            #nextLine
            li $v0,4
            la $a0,newLine
            syscall

            jr $ra
```



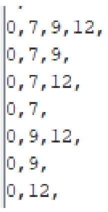
```
0,3,10,11,
0,3,10,
0,3,10,11,
0,3,10,
0,3,7,9,11,
0,3,7,9,
0,3,7,9,11,
0,3,7,9,
0,3,7,11,
0,3,7,
0,3,7,11,
0,3,7,
0,3,9,11,
0,3,9,
0,3,9,11,
0,3,9,
0,3,4,11,
0,3,4,
0,3,11,
0,3,
0,10,11,
0,10,
0,10,11,
0,10,
0,10,11,
0,10,
0,10,11,
0,10,
0,10,11,
0,10,
0,10,11,
0,10,
0,10,11,
0,10,
0,10,11,
0,10,
0,7,9,11,
0,7,9,
0,7,9,11,
0,7,9,
0,7,11,
0,7,
0,7,11,
0,7,
0,9,11,
0,9,
0,9,11,
0,9,
0,4,11,
```

Here I have listed the numbers 3,10,7,9,4,11 about increasing subsequence order. I always print the number 0 at the beginning. Because I did not understand where the number 0 came from, I could not fix that part. I also print all the sequences in between with the longest series.



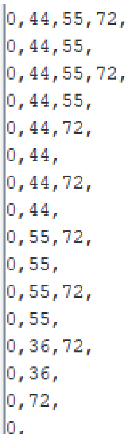
```
0,28,35,102,
0,28,35,102,
0,28,35,50,
0,28,35,
0,28,35,102,
0,28,35,102,
0,28,35,50,
0,28,35,
0,28,102,
0,28,102,
0,28,50,
0,28,
0,28,102,
0,28,102,
0,28,50,
0,28,
0,35,102,
0,35,102,
0,35,50,
0,35,
0,35,102,
0,35,102,
0,35,50,
0,35,
0,3,102,
0,3,102,
0,3,50,
0,3,
0,102,
0,102,
0,50,
0,
```

I ordered the 28,35,3,102,50 ordinal numbers above according to the rule (increasing). I also print all the sequences in between with the longest series.



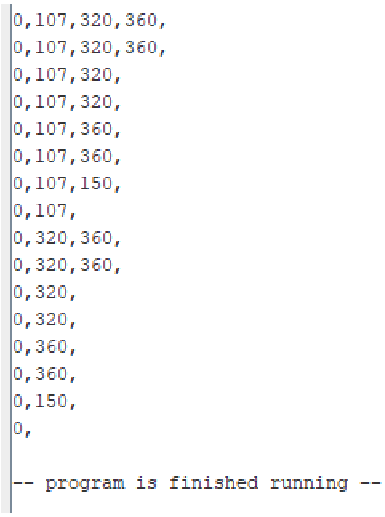
```
 .
0,7,9,12,
0,7,9,
0,7,12,
0,7,
0,9,12,
0,9,
0,12,
```

I ordered the 7,9,12 ordinal numbers above according to the rule (increasing). I also print all the sequences in between with the longest series.



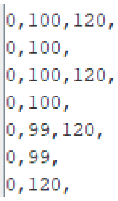
```
0,44,55,72,
0,44,55,
0,44,55,72,
0,44,55,
0,44,72,
0,44,
0,44,72,
0,44,
0,55,72,
0,55,
0,55,72,
0,55,
0,36,72,
0,36,
0,72,
0.
```

I ordered the 44,55,36,72 ordinal numbers above according to the rule (increasing). I also print all the sequences in between with the longest series.



```
0,107,320,360,
0,107,320,360,
0,107,320,
0,107,320,
0,107,360,
0,107,360,
0,107,150,
0,107,
0,320,360,
0,320,360,
0,320,
0,320,
0,360,
0,360,
0,150,
0,

-- program is finished running --
```

I ordered the 107,320,360,150 ordinal numbers above according to the rule (increasing). I also print all the sequences in between with the longest series.



```
0,100,120,
0,100,
0,100,120,
0,100,
0,99,120,
0,99,
0,120,
```

I ordered the 100,99,120 ordinal numbers above according to the rule (increasing). I also print all the sequences in between with the longest series.

Bonus Part: While I was making Increasing Subsequence, I print all the number sequences in between.

Missing Parts: While printing the numbers, my algorithm always prints the number 0 at first, and I did not do the file reading and writing. I adjusted the numbers in the array myself and printed 6 different arrays.