CSE 331/503
Computer Organization
Homework 2 Report

Psuedo Code Of Finding Sequence:

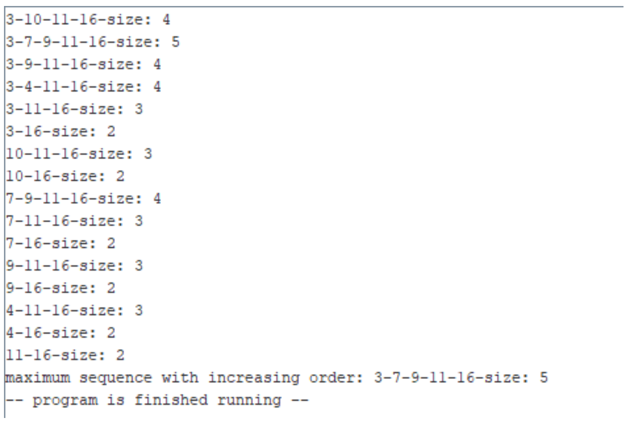
```
for i=0; i<size ; i++
   for j=i+1; j<size; j++
      max = array[i]  --> max variable holds the largest integer.
      save max in temporary array
      if array[j] > max
         max = array[j] --> sequence become array[i], array[j]
         save max in temporary array
         for k = j+1 ; k<size; k++
            if array[k] > max
               max = array[k]
               save max in temporary array

         if temporary array's size > largest sequence array's size
         copy temporary array to largest sequence array
```

Test Cases:

TEST CASE NUMBER	INPUT(From input.txt)	EXPECTED RESULT	PASS/FAIL
1	3,10,7,9,4,11,16	3,7,9,11,16	**PASS**
2	1,2,3,4,5,6,7,8	1,2,3,4,5,6,7,8	**PASS**
3	2,3,1,11,12,13,8	2,3,11,12,13	**PASS**
4	9,8,7,6,5,4,10,12	9,10,12	**PASS**
5	8,12,14,9,10,11,7	8,9,10,11	**PASS**
6	10,7,1,2,3,4,5	1,2,3,4,5	**PASS**

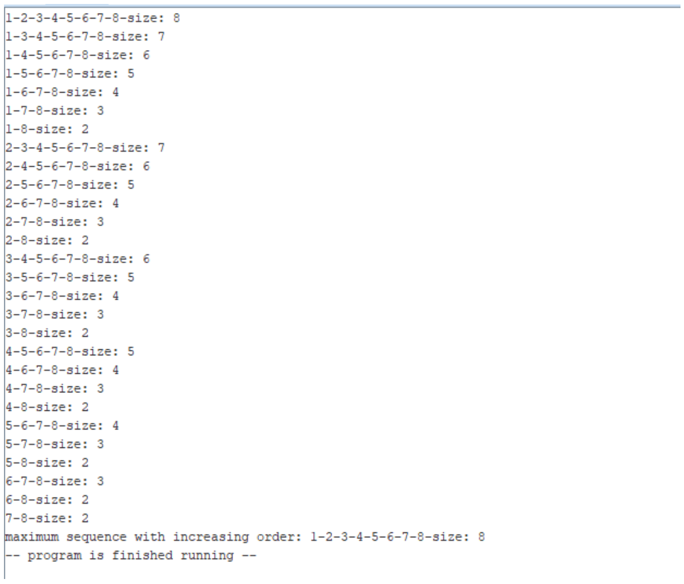
Outputs:

Test Case 1



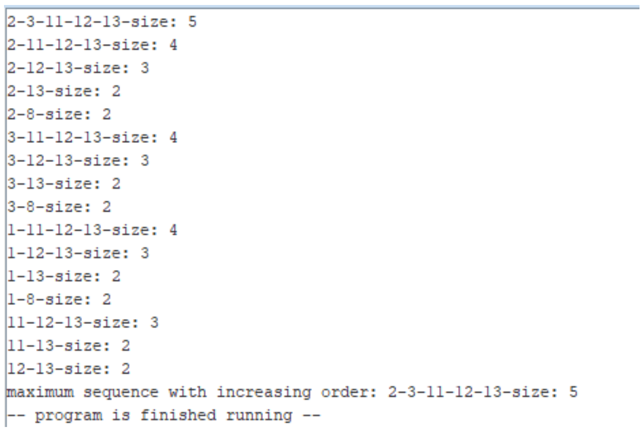
```
3-10-11-16-size: 4
3-7-9-11-16-size: 5
3-9-11-16-size: 4
3-4-11-16-size: 4
3-11-16-size: 3
3-16-size: 2
10-11-16-size: 3
10-16-size: 2
7-9-11-16-size: 4
7-11-16-size: 3
7-16-size: 2
9-11-16-size: 3
9-16-size: 2
4-11-16-size: 3
4-16-size: 2
11-16-size: 2
maximum sequence with increasing order: 3-7-9-11-16-size: 5
-- program is finished running --
```

Test Case 2



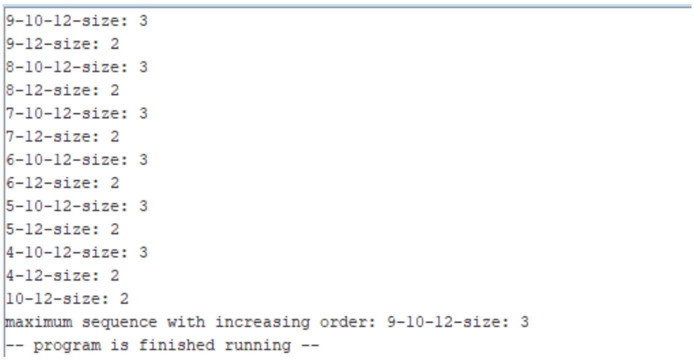
```
1-2-3-4-5-6-7-8-size: 8
1-3-4-5-6-7-8-size: 7
1-4-5-6-7-8-size: 6
1-5-6-7-8-size: 5
1-6-7-8-size: 4
1-7-8-size: 3
1-8-size: 2
2-3-4-5-6-7-8-size: 7
2-4-5-6-7-8-size: 6
2-5-6-7-8-size: 5
2-6-7-8-size: 4
2-7-8-size: 3
2-8-size: 2
3-4-5-6-7-8-size: 6
3-5-6-7-8-size: 5
3-6-7-8-size: 4
3-7-8-size: 3
3-8-size: 2
4-5-6-7-8-size: 5
4-6-7-8-size: 4
4-7-8-size: 3
4-8-size: 2
5-6-7-8-size: 4
5-7-8-size: 3
5-8-size: 2
6-7-8-size: 3
6-8-size: 2
7-8-size: 2
maximum sequence with increasing order: 1-2-3-4-5-6-7-8-size: 8
-- program is finished running --
```

Test Case 3



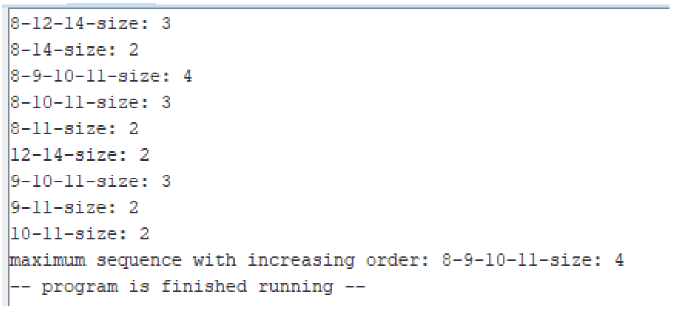
```
2-3-11-12-13-size: 5
2-11-12-13-size: 4
2-12-13-size: 3
2-13-size: 2
2-8-size: 2
3-11-12-13-size: 4
3-12-13-size: 3
3-13-size: 2
3-8-size: 2
1-11-12-13-size: 4
1-12-13-size: 3
1-13-size: 2
1-8-size: 2
11-12-13-size: 3
11-13-size: 2
12-13-size: 2
maximum sequence with increasing order: 2-3-11-12-13-size: 5
-- program is finished running --
```

Test Case 4



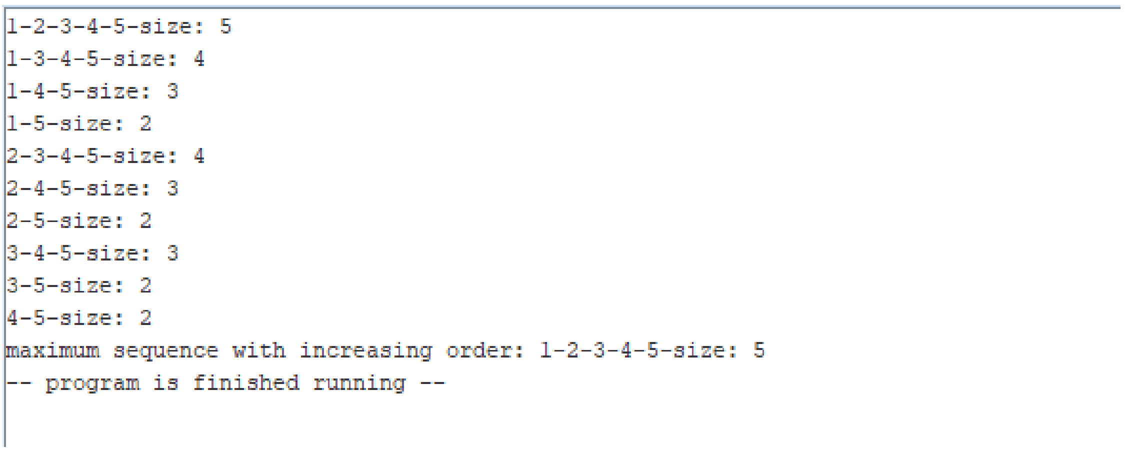
```
9-10-12-size: 3
9-12-size: 2
8-10-12-size: 3
8-12-size: 2
7-10-12-size: 3
7-12-size: 2
6-10-12-size: 3
6-12-size: 2
5-10-12-size: 3
5-12-size: 2
4-10-12-size: 3
4-12-size: 2
10-12-size: 2
maximum sequence with increasing order: 9-10-12-size: 3
-- program is finished running --
```

Test Case 5



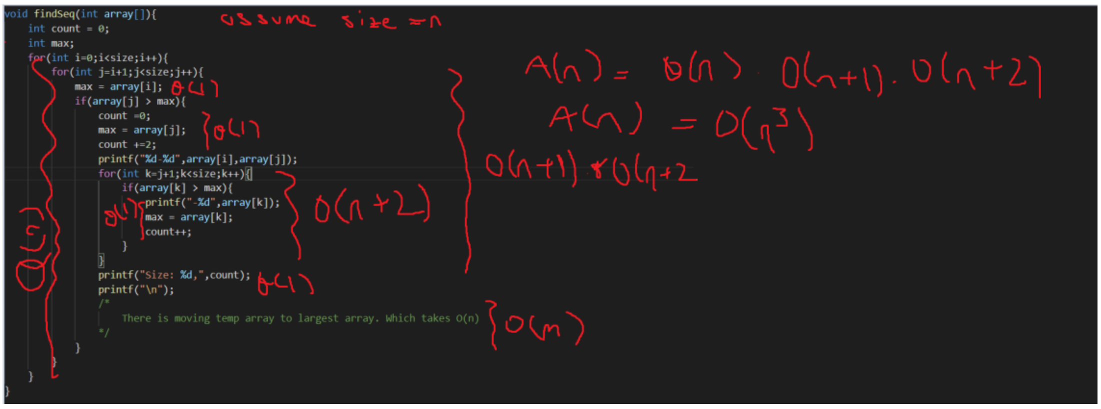
```
8-12-14-size: 3
8-14-size: 2
8-9-10-11-size: 4
8-10-11-size: 3
8-11-size: 2
12-14-size: 2
9-10-11-size: 3
9-11-size: 2
10-11-size: 2
maximum sequence with increasing order: 8-9-10-11-size: 4
-- program is finished running --
```

Test Case 6



```
1-2-3-4-5-size: 5
1-3-4-5-size: 4
1-4-5-size: 3
1-5-size: 2
2-3-4-5-size: 4
2-4-5-size: 3
2-5-size: 2
3-4-5-size: 3
3-5-size: 2
4-5-size: 2
maximum sequence with increasing order: 1-2-3-4-5-size: 5
-- program is finished running --
```

Time Complexity:



```
void findSeq(int array[]){
    int count = 0;
    int max;
    for(int i=0;i<size;i++){
        for(int j=i+1;j<size;j++){
            max = array[i];
            if(array[j] > max){
                count =0;
                max = array[j];
                count +=2;
                printf("%d-%d",array[i],array[j]);
                for(int k=j+1;k<size;k++){
                    if(array[k] > max){
                        printf("-%d",array[k]);
                        max = array[k];
                        count++;
                    }
                }
                printf("Size: %d,",count);
                printf("\n");
                /*
                    There is moving temp array to largest array. Which takes O(n)
                */
            }
        }
    }
}
```

Space Complexity:

2 array → $\Theta(n) + \Theta(n) = A(n) = \Theta(n)$

Functions:

Openfile function open “input.txt”. If there is a error it stores zero in $t0 register. Error checking executes in main.

```
openfile:
    li $v0, 13
    li $a1, 0
    li $a2, 0
    syscall
    move $s0, $v0 #file descriptor in s0 register
    slti $t0, $v0,0
    jr $ra
```

Read function read input and stores into buffer.

```
read:
    li $v0, 14
    move $a0, $s0
    la $a1, buffer
    li $a2, 44
    syscall
    j initialize
```

fillArray function reads buffer until comma encounter and stores numbers into temporary array. atoi load functions store necessary values on registers and call atoi function.

```
fillArray:
    lb $t2, buffer($s0) #loading char to t2 register
    beq $t2, 0, flag1
    beq $t2, 44, atoiload # comma come
    sb $t2, temparray($s1) # loading char to temp array
    addi $s0, $s0, 1
    addi $s1, $s1, 1
    j fillArray
```

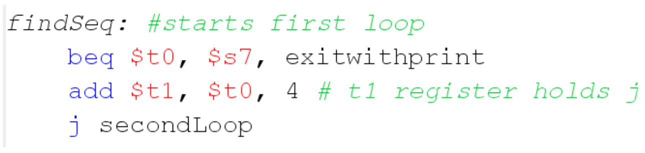
Atoi function reads charachter reversly from temp array and subtratcs 48(0 in ascii) from them to get their value. After that it calculates 10^digit*value and store it to $t3 until the charachters end.

```
atoi:
    beq $t0, $zero, reset
    addi $t0, $t0, -1
    lb $t4 , temparray($t0)
    addi $t4, $t4, -48
    mul $t2, $t1, $t4
    add $t3, $t3, $t2
    mul $t1, $t1, 10
    j atoi
```

init function initiliaze neccessary register which will be used for finding sequence

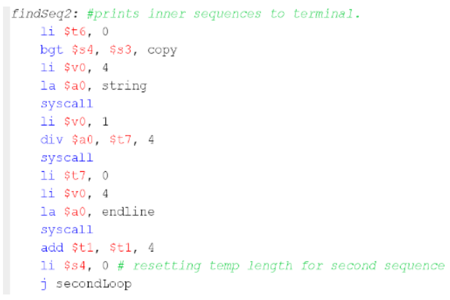
```
#finding sequence
init: # initializes neccessary values
    li $t7, 0 #counter
    li $t0, 0 #i
    li $s3, 0 # longest length
    li $s4, 0 # temp length
```

findSeq function starts first loop in psuedo code



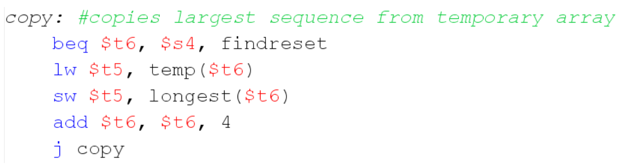
```
findSeq: #starts first loop
    beq $t0, $s7, exitwithprint
    add $t1, $t0, 4 # t1 register holds j
    j secondLoop
```

findSeq2 function prints inner sequence to terminal



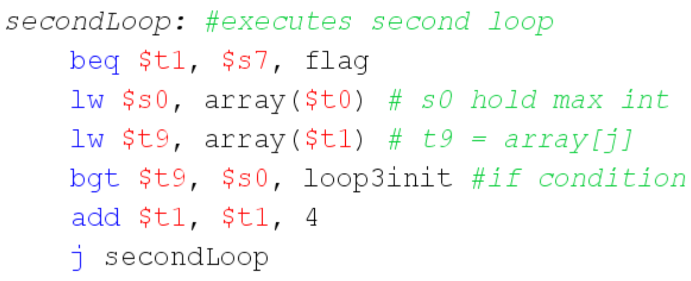
```
findSeq2: #prints inner sequences to terminal.
    li $t6, 0
    bgt $s4, $s3, copy
    li $v0, 4
    la $a0, string
    syscall
    li $v0, 1
    div $a0, $t7, 4
    syscall
    li $t7, 0
    li $v0, 4
    la $a0, endline
    syscall
    add $t1, $t1, 4
    li $s4, 0 # resetting temp length for second sequence
    j secondLoop
```

Copy function copies longest sequence to longest array from temporary array.



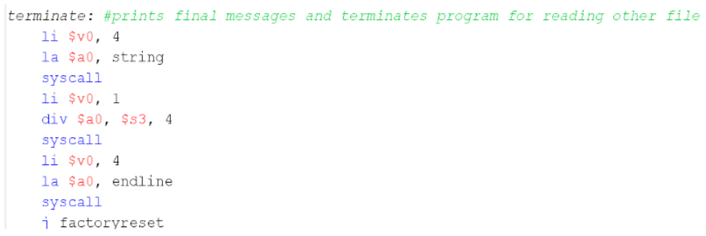
```
copy: #copies largest sequence from temporary array
    beq $t6, $s4, findreset
    lw $t5, temp($t6)
    sw $t5, longest($t6)
    add $t6, $t6, 4
    j copy
```

secondLoop function executes second loop in psuedo code.



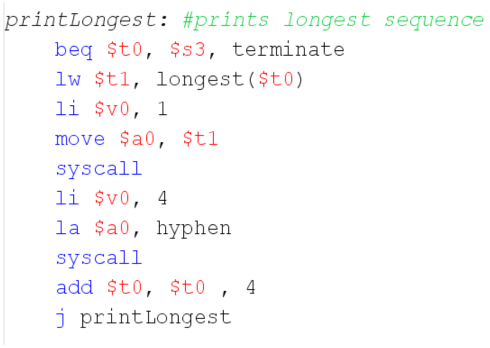
```
secondLoop: #executes second loop
    beq $t1, $s7, flag
    lw $s0, array($t0) # s0 hold max int
    lw $t9, array($t1) # t9 = array[j]
    bgt $t9, $s0, loop3init #if condition
    add $t1, $t1, 4
    j secondLoop
```

Terminate function prints final message and terminate program for reading other file



```
terminate: #prints final messages and terminates program for reading other file
    li $v0, 4
    la $a0, string
    syscall
    li $v0, 1
    div $a0, $s3, 4
    syscall
    li $v0, 4
    la $a0, endline
    syscall
    j factoryreset
```

printLongest function prints longest sequence.



```
printLongest: #prints longest sequence
    beq $t0, $s3, terminate
    lw $t1, longest($t0)
    li $v0, 1
    move $a0, $t1
    syscall
    li $v0, 4
    la $a0, hyphen
    syscall
    add $t0, $t0 , 4
    j printLongest
```