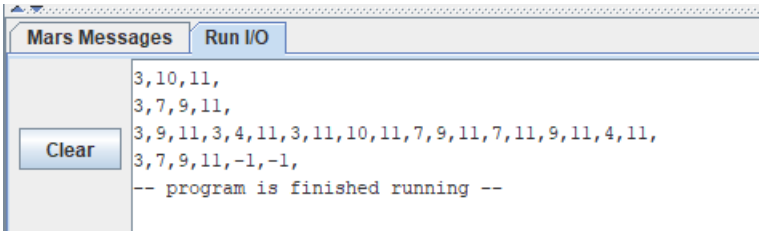
Test Cases

Test case for array {3, 10, 7, 9, 4, 11}

Last row is the longest increasing subsequence, minus ones are there because of a bug.



Explanation

```
for1:   beq $s0 , $s6 , out1 # loop until i == size
        add $s1 , $s0 , $zero # j = i

for2:   beq $s1 , $s6, out2  # loop until j == size
        move $s3 , $zero # counter = 0

        # if arr[j] > arr[i]
        mul $t0 , $s1 , 4
        mul $t9 , $s0 , 4
        lw $t1 , array($t0) # arr[j] is $t1
        lw $t2 , array($t9) # arr[i] is $t2
        slt $t3 , $t2 , $t1 # if arr[i] < arr[j] t3 is 1 else t3 is 0
        addi $s1 , $s1 , 1 # j++ to return to for2
        beq $t3 , $zero , for2

        addi $s1 , $s1 , -1 # j--

        li $v0 , 1 # printf("%d , %d ,", arr[i] , arr[j]);
        move $a0 , $t2
        syscall
        li $v0 , 4
        la $a0 , coma
        syscall
        li $v0 , 1
        move $a0 , $t1
        syscall
        li $v0 , 4
        la $a0 , coma
        syscall

        add $s2 , $s1 , $zero # k = j
for3:   beq $s2 , $s6 , out3 # loop until k == size
        #if arr[k] > arr[j]

        mul $t9 , $s2 , 4 # 4k
        lw $t4 , array($t9) # arr[k] is $t4
        mul $t8 , $s1 , 4
        lw $t1 , array($t8)
        slt $t3 , $t1 , $t4 # if arr[j] < arr[k] t3 is 1 else t3 is 0
        addi $s2 , $s2 , 1 # k++ to return to for3
```

```
beq $t3 , $zero , for3

li $v0 , 1 # printf("%d ,", arr[k]);
move $a0 , $t4
syscall
li $v0 , 4
la $a0 , coma
syscall

addi $s3 , $s3 , 1 # counter++
j for3 # continue to loop for3
```

To scan entire array I use for1 , for2 ,for3 which are I , j and k. Without “k” , finding the all candidates is impossible , because “I” and “j” is busy to scan another value.

Missing parts

I didn’t test my program for 6 different arrays.

Program has a bug in printing the all longest increasing subsequence candidate arrays.

When printing the longest increasing subsequence , its size is broken so I had to print the entire array including garbage values (-1).

Time Complexity

```c
#include <stdio.h>

void main(){
    int arr[6] = {3, 10, 7, 9, 4, 11};
    int temp[6];
    int counter = 0 , max = 0 , index_cntr = 0;

    for(int i = 0; i< 6 ; i++){

        for(int j = i ; j< 6; j++){
            counter = 0;
            if( arr[j] > arr[i]){
                printf("%d , %d ,", arr[i] , arr[j]);

                for(int k = j ; k< 6 ; k++){
                    if(arr[k] > arr[j]){
                        printf("%d , ", arr[k]);
                        counter++;
                    }

                }
                if(max < counter){
                    max = counter;
                    index_cntr = 0;
                    temp[index_cntr++] = arr[i];
                    temp[index_cntr++] = arr[j];

                    for(int k = j ; k< 6 ; k++){
                        if(arr[k] > arr[j]){
                            temp[index_cntr++] = arr[k];
                        }
                    }
                }
                printf("\n");
            }

        }
    }
```

```c
    for(int i = 0; i< index_cntr; i++){
        printf("%d , ",temp[i]);
    }
    printf("size : %d ",index_cntr);
}
```

For the most inner loop , time complexity is O(n) for worst case.

For the inner loop , time complexity is O(n^2) for worst case.

For the most outer loop, time complexity is O(n^3) for worst case.

For the bottom loop , time complexity is O(n) for worst case. But it doesn’t matter because the entire time complexity will be O(n^3 + n) , so n is neglicible.

Entire program’s time complexity is O(n^3).