1.Pseudo Code

```
for(i := 0 to sizeof Array )
{
    Set counter_temp 0;
    Set temp[counter_temp] = arr[i];

        for(j = i+1 ; to sizeof Array)
        {
                If temp[counter_temp]<arr[j]
                    Then set temp[++counter_temp] = arr[j];

                    for(m = j+1: to higher than sizeof Array)
                    {
                        if(temp[counter_temp]<arr[m])
                            then
                             Increase counter_temp by 1
                             Set tempArr = main Arr[m]

                    }

                if(counter_temp+1 is higher than counter )
                {
                 Set counter = counter_temp+1;

                for( h = 0; to h  larger than counter ; h++)
                   set last[h] = temp[h]; //Last array is the final array
                }

            Set counter_temp = 0; // End of every loop
        }
}
```

For_i Function:

- First loop only equalize with temporary counter which is turns 0 in every loop, and temprorary array initialize with element to compare

```
for_i:
        bgt  $t2, $t8, exit # if (i > sizeofarray , go exit)
        addi $t0, $zero, 0 # counterTemp = 0

        lw $t9, myArray($t2)
        sw $t9, myTempArr($t0)  # myTemparr[counterTemp] = myArray[i];
        addi $t2, $t2, 4 # counter i++


        addi $t6, $t2, 0 # $t6 = j and j = i+1
        j for_j
```

```
for(i = 0; i<6 ; i++)
{
    counter_temp = 0;
temp[counter_temp] = arr[i];

        for(j = i+1 ; j<6; j++) { ... }
}
```

For_j Function :

- In this loop compares the previous (which is related to i) element of the array with the element in the iterator if iterator index is higher than previous element. The value of in the array[iterator] function copies the value to temp array

```
for_j:
        bgt $t6, $t8, for_i # if j > sizeofArray go forI
        #addi $t0, $t0, 4 # counterTemp = size will increase 4 counter temp++

        lw $t3, myTempArr($t0)
        lw $t4, myArray($t6)
        blt $t3,$t4,equalize
        bge $t3, $t4, return
return:
        addi $t6, $t6, 4 # j++

        addi $t5, $t6, 0 # $t5 = k and k = j+1
        j for_k
```

```
        for(j = i+1 ; j<6; j++)
        {
                if(temp[counter_temp]<arr[j])
                    temp[++counter_temp] = arr[j];

                    for(int m = j+1; m<6 ; m++) { ... }

                if(counter_temp+1 > counter )
                {
                //print_arr(temp,counter_temp);
                //printf("Counter : %d\n",counter_temp+1);
                counter = counter_temp+1;

                for(int h = 0; h<counter ; h++)
                    last[h] = temp[h];
                }

            counter_temp = 0;
        }
```

For_k Function :

This function provides to help us to find all subsequence of array with ascending order by comparing them with the second iterator index of array which is j then it prints subsequences of array immediately in this for loop.

```
for(int m = j+1; m<6 ; m++)
{
    if(temp[counter_temp]<arr[m])
        {
        counter_temp +=1;
        temp[counter_temp] = arr[m];
        print_arr(temp,counter_temp+1);
        }
}
```

```
for_k:

        bgt $t5, $t8, for_j

        lw $t3, myTempArr($t0)
        lw $t4, myArray($t5)

        blt $t3, $t4, equal
        bgt $t3, $t4,return_back
return_back:
        addi $t5, $t5, 4
        j for_k

equal:
        addi $t0, $t0, 4
        lw $t9, myArray($t5)
        sw $t9, myTempArr($t0)
```

1.Test Case Input : {3,10,7,9,4,11}

In Mars Simulator Test	In C test
-- program is finished running (dropped off bottom) -- 31011 101111 7911 91111 0	clang-7 -pthread -lm -o main main.c ./main 3,10,11, 3,7,9, 3,7,9,11, 3,9,11, 3,4,11, 10,11, 10,11, 10,11, 7,9,11, 7,11, 9,11, Size of array : 4 {3,7,9,11, }

2.Test Case Input : {3,10,5,7,9,4,11,12}

In Mars Simulator Test	In C test
31011 101111 101112 7911 7911 91111 91112 Clear	10,11,12, 10,11,12, 5,7,9, 5,7,9,11, 5,7,9,11,12, 5,9,11, 5,9,11,12, 5,11, 5,11,12, 5,11,12, 7,9,11, 7,9,11,12, 7,11, 7,11,12, 7,11,12, 9,11, 9,11,12, 9,11,12, 4,11,12, Size of array : 6 {3,5,7,9,11,12, }

3.1 Time Complexity n . (n-1) . (n-2)

- **Each for loop index will increase by 1 and for loop return 1 less in every loop**

3.2 Space Complexity

- **Main Array of n**
- **Temp array size of n**
- **Last printing array size of n**
- **The space complexity of algorithm is 3n**

4. Missing Parts

- Reading From file
- Write to file
- Converting char to integer
- Converting 2 separate char to a 2 digit integer

Note :

Reading from file and converting from char to integer part will done by following code but I didn't put it in the homework because it didn't make any sense for the homework algorithm. I didn't put this file in homework because it didn't make any sense for the homework. I put the part of reading from the file I wrote and converting to integer in the photo below.

```
.data

filename : .asciiz "test.txt"
buffer : .space 40
charZero: .ascii "0"
minus4: .word -4



.text

    addi $t7, $zero, -4
    addi $t6, $zero, 10
    #open a file for reading purpose
    li $v0, 13    # system call for open file
    la $a0, filename    # input file name
    li $a1, 0    # open for reading (flags are 0: read, 1: write)
    li $a2, 0    # mode is ignored
    syscall        # open a file (file descriptor returned in $v0)
    move $s6, $v0    # save the file descriptor
##########################################################################



    while:
         beq $t1, -35, exit

    ##########################################

         li $v0,14    # system call for read from file
         move $a0,$s6    # file descriptor
         la $a1,buffer    # address of buffer from which to read
         li $a2,1   # hardcoded maximum number of chars to read
         syscall        # read from file

    ##################################################


         lb $t2,charZero
         lb $t1,buffer
         sub $t1,$t1,$t2

        #jr $ra

         j while

##############################################################################

#################################################################################
  exit:
       # close the file
     li $v0,16     # system call for close file
     move $a0,$s6     # file descriptor to close
     syscall          # close file
##############################################################################

     li $v0,10
     syscall
```