CSE 331 Computer Organization Homework 1

Yakup Talha Yolcu 1801042609

1. C CODE

```c
#include <stdio.h>
void main() {
    int arr[]={3,10,7,9,4,11};
    int arr_size=6;
    int out2[arr_size];
    int counter_old=0;
    printf("\n");
    for(int i=0;i<arr_size;i++) {
        int max_v=arr[i];
        int out[arr_size];
        int counter=0;
        if(i!=0) {
            //look back
            int max_of_array=0;
            for(int j=0;j<i;j++) {
                if(arr[j]<max_v && max_of_array<arr[j]) {
                    out[counter]=arr[j];
                    counter++;
                    max_of_array=arr[j];
                }
            }
        }
        out[counter]=arr[i];
        counter++;
        for(int j=i+1;j<arr_size;j++) {
            if(max_v<arr[j]) {
                out[counter]=arr[j];
                max_v=arr[j];
                counter++;
            }
        }
        for(int z=0;z<counter;z++) {
            printf("%d ",out[z]);
        }
        printf("\n");
        if(counter_old<counter) {
            for(int z=0;z<arr_size;z++) {
                out2[z]=0;
            }
            for(int z=0;z<counter;z++) {
                out2[z]=out[z];
            }
            counter_old=counter;
        }
    }
    printf("Result is:\n");
    for(int z=0;z<counter_old;z++) {
            printf("%d ",out2[z]);
    }
}
```

Input array:arr

We know array_size

We have output array named out2

For loop looks all the elements of the array

Max_v holds the current element of the array at the beginning

We should declare temp array which named out. We have **counter that stores the temp arrays length**

In my algorithm I check first we are at the first element in the array or not. If we are in not in the first element, then we should look back to the get less elements than current element.

Max_of_array is 0 in the beginning

Then we are looking back, finding a less element and store it into temp array. **Max of array** becomes maximum of the **tempArr** until now.

End of for loop, we are adding current element to temp array and increment counter

Next, we are looking the rest of the array and find greater element and add to the temp Array. We should check max_v because we don't want to have less element in here.

2. PSEUDOCODE

```
procedure(arr[0:n-1],int length) {
    declare output array with size length
    initialize counter_old to 0
    for i=0,i<arr_size,i++
        initialize max_v to arr[i]
        declare temp array with size length
        initialize counter to 0
        if i!0 then
            //look previous elements
            initialize max_of_array to 0
            for j=0,j<i,j++
                if arr[j]<max_v and max_of_array<arr[j] then
                    set output[counter] to arr[j]
                    increment counter by 1
                    set max_of_array to arr[j]
                endif
            endfor
        endif
        set output[counter] to arr[i]
        increment counter by 1
        for j=i+1,j<arr_size,j++
            if max_v<arr[j] then
                set output[counter] to arr[j]
                set max_v to arr[j]
                increment counter by 1
            endif
        endfor
        if counter_old<counter  then
            this means that we have found a longer sequence
            firstly clear output array
            then copy temp array to output array
            set counter_old to counter
        endif
    endfor
}
```

This part is **constant time**

This part is **constant time**

This part is **constant time**

Average time complexity of here is **linear time** because at the worst case(last index) we should look all the array

Constant time

In this for loop, we are looking the rest of the array, so this means that average time complexity is **linear time** because in the worst case we are looking all the elements in the array.

Clearing output array is linear time

Copying arrays is **linear time**

Setting is **constant time**

All operations above will be executed n times, so this means that our space complexity is $\theta(n^2)$

3. SPACE COMPLEXITY AND TIME COMPLEXITY

As I mentioned, Time complexity of the algorithm is **$\theta(n^2)$** . Space complexity is **n** because we are just having 3 array with length n and we are clearing them.

4. EXPLANATIONS

```
52
53  main:
54
55          #open file for reading
56          li $v0,13               #system call for open file
57          la $a0,file_name        #input file name
58          li $a1,0                #flag for reading
59          li $a2,0                #mode is ignored
60          syscall                 #open a file
61          move $s0,$v0            #save the file descriptor
62
63          # reading from file just opened
64          li   $v0, 14       # system call for reading from file
65          move $a0, $s0      # file descriptor
66          la   $a1, buffer   # address of buffer from which to read
67
68          li   $a2,  240             # hardcoded buffer length
69          syscall            # read from file
70
71          #print the buffer
72          li $v0,4
73          la $a0,buffer
74          syscall
75
76          jal close_the_read_file
77
```

Here I just opened file and read buffer, then I printed the buffer to see what I got

```
        # Open (for writing) a file that does not exist
        li $v0,13       #system call for open file
        la $a0,fout     #output file name
        li $a1,1        #open for writing flags are 0: read 1:write
        li $a2,0        #mode is ignored
        syscall         #open a file file descriptor returned in $v0

        move $s6,$v0


        #write to file just opened
        li $v0,15       #system call for write to file
        move $a0,$s6    #file descriptor
        la $a1,newLine  #adress of buffer from which to write
        li $a2,1        #hardcoded buffer length
        syscall         #write to file

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




        j start_the_process
```

Here I just want to make sure that output.txt is empty, I opened to write it and then I directly close it.

```
106
107 start_the_process:
108
109         #excalamation gorene kadar devam edecez
110         #her exclamationda döngü olacak
111         #6.exclamationdan sonra da bu döngüden çıkacaz
112
113         addi $t0,$zero,0        #exclamation counter
114         addi $t1,$zero,0        #buffer cursor  holds the end index of the array in the buffer
115
116
117 process_loop1:
118         beq $t0,6,exit_process  #if(t0==6 then exit from the process, all arrays are evaluated write file should be closed)
119         add $t4,$zero,$t1       #t4 holds the start index of the array in the buffer
120         #find_the_length_of_array
121         #addi $t1,$zero,0               #counter
122         find_length_of_buffer_loop:
123                 lb $t2,buffer($t1)              #t2=buffer[counter]
124                 lb $t3,exclamation($zero)       #t3='!'
125                 beq $t2,$t3,find_length_of_buffer_exit_loop             #if t2==t3 then exit loop
126                 addi $t1,$t1,1                  #else increment counter
127                 j find_length_of_buffer_loop
128
129         find_length_of_buffer_exit_loop:
130
131
```

T0 register holds the number of arrays evaluated.

T1 register holds the last index of the current array in the buffer.

T2 holds the buffer[counter]

T3 holds the exclamation

I compare them and if it is exclamation then I should exit from that loop

If it is not excalamation then I continue until excalamation.

Finally I fount the length of array

save_number_into_buffer1_loop1 line 139:

In this function I save the integer numbers into memory from getting characters and use atoi function.

outer_loop line 232:

In this function , I am iterating through every element of the integer array. This is my algorithm to detect the longest sequence.

look_back line 244:

In this function this means we are not in the first element, we need to look back because we could find less elements than current element.

don_t_look_back line 288:

In this function we are finding greater element than current element at the rest of the array.

go_into_third_if line 325:

In this function we are checking that temp sequence is longer or not our previous sequence. If it is then we are updating our sequence

```
379
380         #write_to_file
381
382
383         #open file
384         li $v0,13       #system call for open file
385         la $a0,fout     #oputput file
386         li $a1,9        #write to file append mode
387         li $a2,0        #ignore mode
388         syscall
389
390         move $s6,$v0    #file descriptor is in the s6
391         move $a3,$s6    #file descriptor is in the v0
392
```

In here I open the file to write in append mode. Append mode is happened when a1 is 9

loop_parse_buffer3 line 401:

In this function I am parsing the integer array to get char array to write the file.

parsed_buffer3: line 429

If we reached this function, our number is parsed properly.

reverse_loop_buffer3 line 439:

We are checking that our number has more than 1 digit or not because if it has more than 1 digit we need to reverse the string to write it to file properly.

exit_loop_parse_buffer3 line 496:

All numbers are parsed, we can write char array to the file

5. RESULT OF TEST CASES

TEST CASE 1
Input array: 3 10 7 9 4 11
Output array: 3 7 9 11

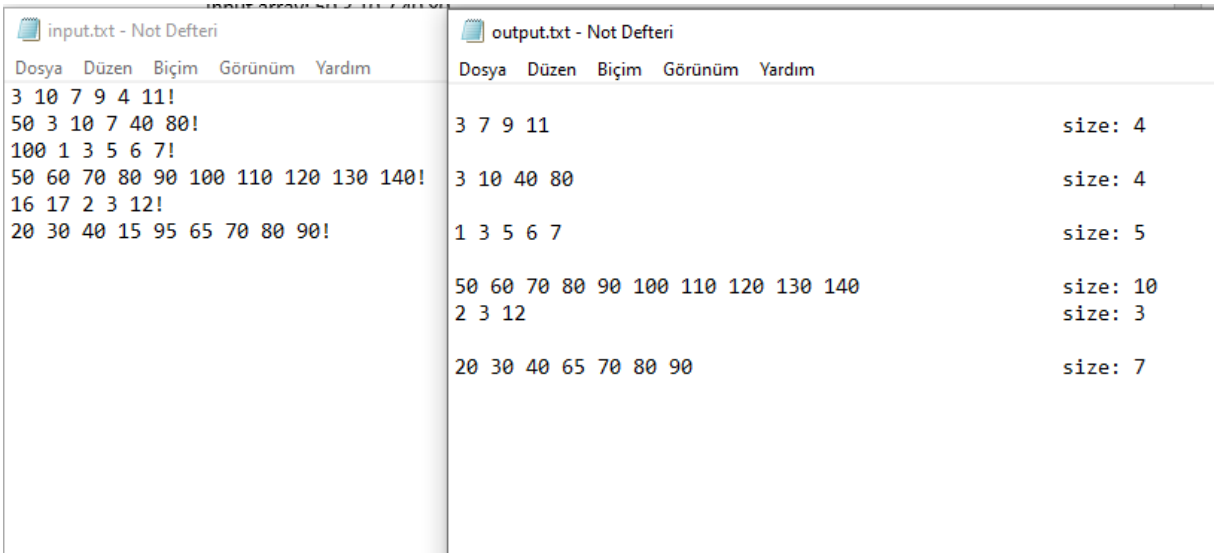
TEST CASE 2
Input array: 50 3 10 7 40 80
Output array: 3 10 40 80

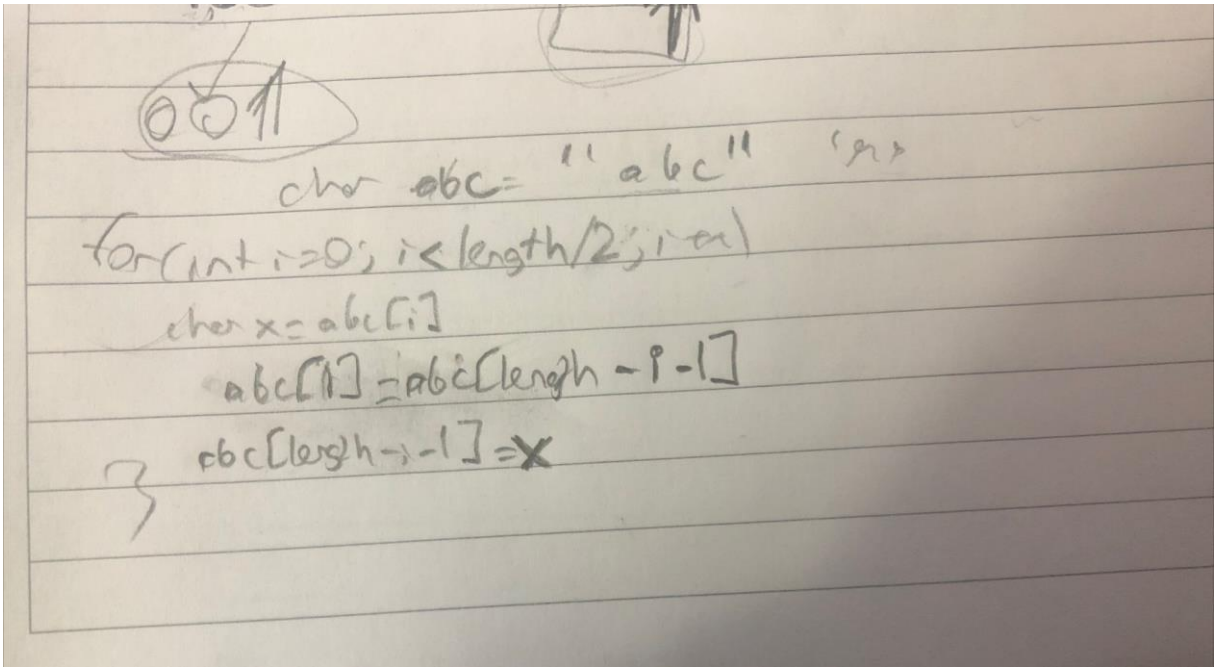
TEST CASE 3
Input array: 100 1 3 5 6 7
Output array: 1 3 5 6 7

TEST CASE 4
Input array: 50 60 70 80 90 100 110 120 130 140
Output array: 50 60 70 80 90 100 110 120 130 140

TEST CASE 5
Input array: 16 17 2 3 12
Output array: 2 3 12

TEST CASE 6
Input array: 20 30 40 15 95 65 70 80 90
Output array: 20 30 40 65 70 80 90





My reverse characters algorithm

```
int myAtoi(char* str)
{
    // Initialize result
    int res = 0;

    // Iterate through all characters
    // of input string and update result
    // take ASCII character of corresponding digit and
    // subtract the code from '0' to get numerical
    // value and multiply res by 10 to shuffle
    // digits left to update running total
    for (int i = 0; str[i] != '\0'; ++i)
        res = res * 10 + str[i] - '0';

    // return result.
    return res;
}
```

My atoi algorithm. I have also itoa algorithm but it does reverse of this operations