BSc (Hons) in Information Technology

Year 2

Foundations of Algorithms – IE2072

IT19001180
Rodrigo K. A. M.

Question 1

```c
//IT19001180
//Rodrigo K. A. M.

#include <stdio.h>

int main()
{
    int n;   //array size
    int m;   //the searching element
    int index = -1;   //initialize the variable index to -1

    printf ("\n");
    printf ("-------------------------------------------------------------\n");
    printf ("\n");

    printf ("Enter a number for n: ");
    scanf ("%d", &n);   //get a number as the size of the array

    printf ("\n");
    printf ("-------------------------------------------------------------\n");
    printf ("\n");

    int ar[n];   //create the array & set the variable n as the array size

    for (int i = 0; i < n; i++){   //input elements to the array
        printf ("Enter a number for the array: ");
        scanf ("%d", &ar[i]);
    }

    printf ("\n");
    printf ("-------------------------------------------------------------\n");
    printf ("\n");
-- INSERT --                                                    19,19         Top
```

```c
    printf ("--- The array, you have created is below ---\n");

    for (int i = 0; i < n; i++){   //display the array that has been created
        printf ("%d ", ar[i]);
    }

    printf ("\n");
    printf ("\n");
    printf ("-------------------------------------------------------------\n");
    printf ("\n");

    printf ("Enter the number you are searching: ");
    scanf ("%d", &m);   //enter the searching element

    printf ("\n");
    printf ("-------------------------------------------------------------\n");
    printf ("\n");

    for (int r = 0; r < n; r++){
        if (m == ar[r])   //searching whether the searching element is in the array or not
            index = r;   //if the searching element found in the array, set that element index as the variable called index
    }


    if (index == -1)   //check whether the index == -1
        printf ("%d is not in the array. Index = %d", m, index);   //print index = -1

    else
        printf ("Index of the value that you searched is %d", index+1);   //print the index

-- INSERT --                                                    60,5          64%
```

```
    scanf ("%d", &m);   //enter the searching element

    printf ("\n");
    printf ("------------------------------------------------------------\n");
    printf ("\n");

    for (int r = 0; r < n; r++){
        if (m == ar[r])   //searching whether the searching element is in the array or not
            index = r;   //if the searching element found in the array, set that element index as the variable called index
    }


    if (index == -1)   //check whether the index == -1
       printf ("%d is not in the array. Index = %d", m, index);   //print index = -1


    else
       printf ("Index of the value that you searched is %d", index+1);   //print the index


    printf ("\n");
    printf ("\n");
    printf ("------------------------------------------------------------\n");
    printf ("\n");

    return 0;
}
```

In here, an array has been created with n elements. So that any number of elements can be input to the array. Plus, there is a variable called m to search the element that we want to find in the array.

The first for loop of the code prints the array. This code segment was written for the clarity of the code.

Since an array has been used in this code when we are going to print the index (this is not relevant when the searching element, the variable m is not in the array), we have to increment the index by 1. Because in arrays the array indexes are starting from 0. That is why the index has been incremented by 1 in below code segment.

```
else

    printf ("Index of the value that you searched is %d", index+1);   //print the index
```

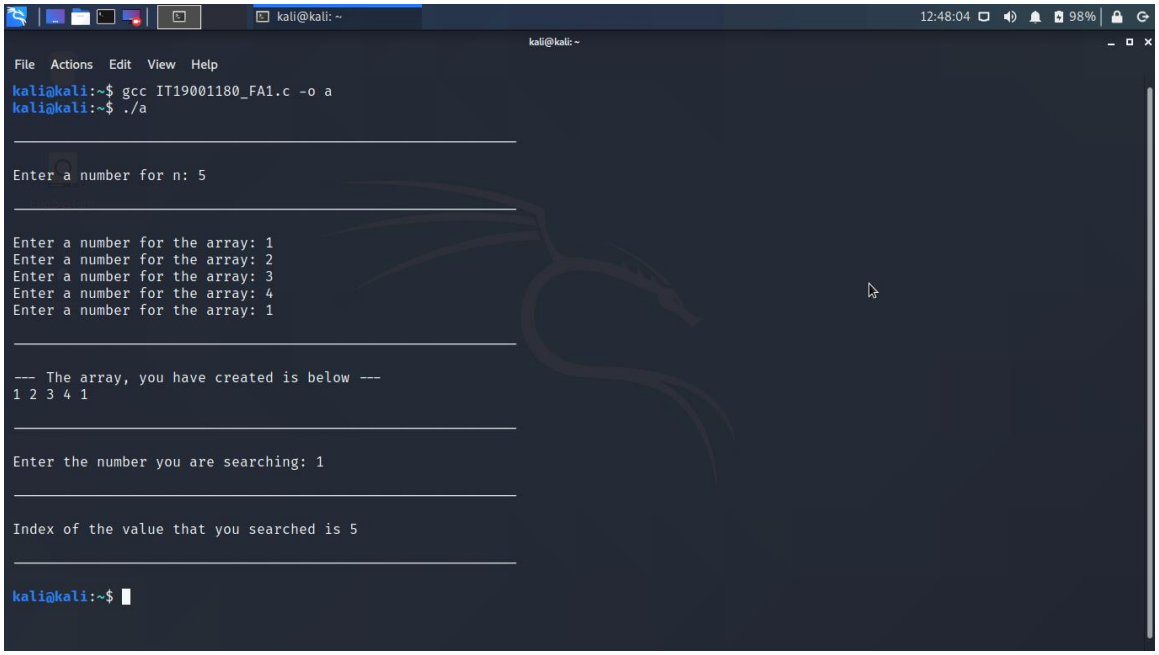


Figure 1: Output 1

Let's turned into the compiled code. In here I have inserted 5 as the size of the array and 1, 2, 3, 4 and 1 as the elements. After that I entered 1 as the element which is going to be searched (variable m). According to the array that has been created there are 2 indexes which contains 1 and we want to find the last occurrence of 1 (variable m). So, the index of last occurrence of 1 is 5. (figure 1)

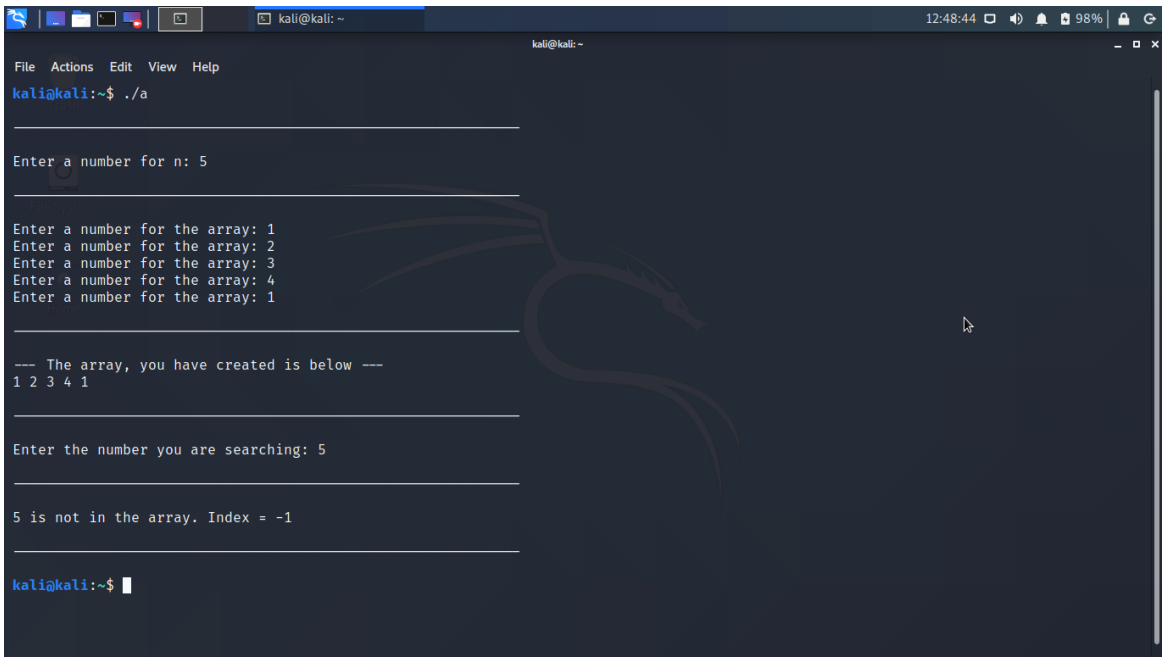


Figure 2: Output 2

If the same code is compiled again with 1, 2, 3, 4 and 1 as the elements and the searching element (variable m) is 5. In this scenario 5 is not in the array. So, the index should be -1. (figure 2)

Question 2

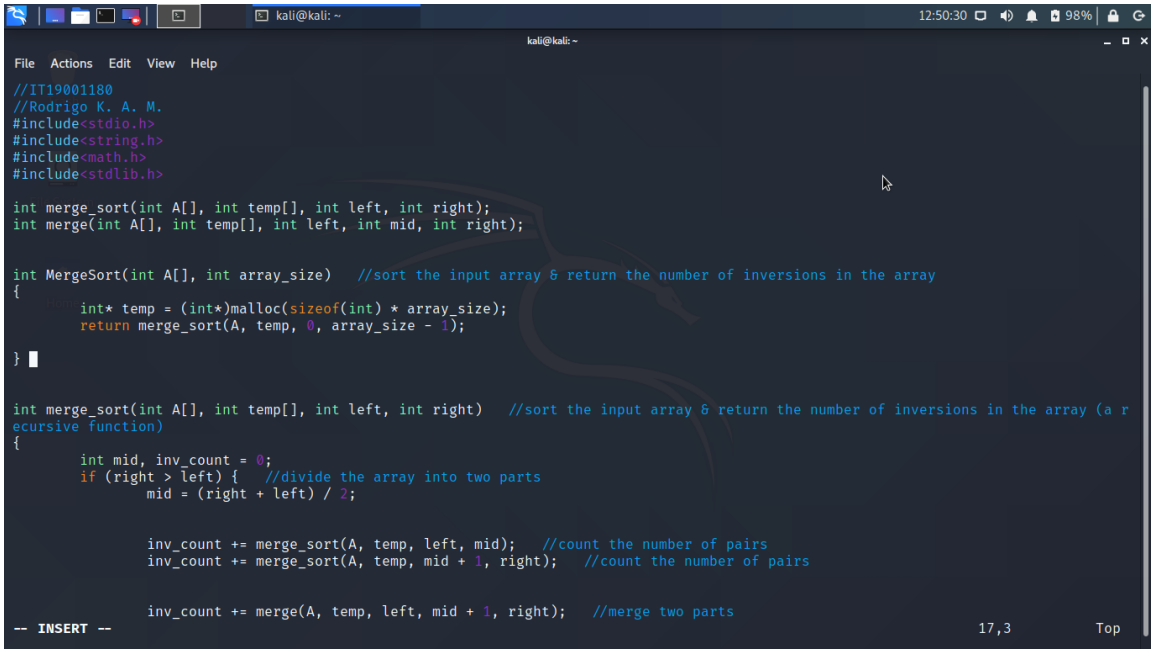
Since we are advised to use divide and conquer method to solve this question, I selected merge sort to do this.

In here three functions are implemented. They are merge_sort, merge and MergeSort.

MergeSort function sorts the input array & returns the number of inversions in the array.

merge_sort function is a recursive function and sorts the input array & returns the number of inversions in the array.

merge function merges the sorted arrays & return the number of pairs.



```
        }
        return inv_count;
}


int merge(int A[], int temp[], int left, int mid, int right)   //merge the sorted arrays & return the number of pairs
{
        int i, j, k;
        int inv_count = 0;

        i = left;   //index for left sub array
        j = mid;   //index for right sub array
        k = left;   //index for merged sub array
        while ((i <= mid - 1) && (j <= right)) {
                if (A[i] <= A[j]) {
                        temp[k++] = A[i++];
                }
                else {
                        temp[k] = A[j];

                        for(int x = i;x<mid; x++)
                                printf("\n(%d,%d)", A[x],temp[k]);


                        inv_count = inv_count + (mid - i);


                        k++;
                        j++;
                }
        }

-- INSERT --                                                                    58,4-25       38%
```

```
        while (i <= mid - 1)   //copy the remaining elements of left sub array to temp*
                temp[k++] = A[i++];


        while (j <= right)   //copy the remaining elements of right sub array to temp*
                temp[k++] = A[j++];


        for (i = left; i <= right; i++)   //copy the merged elements to the original array
                A[i] = temp[i];

        return inv_count;
}


int main(int argv, char** args)
{
        int n;   //create the variable n

    printf ("------------------------------------------------------------------------");
    printf ("\n");
    printf ("\n");
    printf ("Enter a number for n: ");   //getting input to set the array size
    scanf ("%d", &n);
    printf ("\n");
    printf ("------------------------------------------------------------------------");

    int A[n];   //create the array & set the variable n as the array size

    printf ("\n");
    printf ("\n");

-- INSERT --                                                                    91,5          79%
```

```
kali@kali: ~                                                    12:52:47  98%
File  Actions  Edit  View  Help
        int n;   //create the variable n

    printf ("------------------------------------------------------------------");
    printf ("\n");
    printf ("\n");
    printf ("Enter a number for n: ");   //getting input to set the array size
    scanf ("%d", &n);
    printf ("\n");
    printf ("------------------------------------------------------------------");

    int A[n];   //create the array & set the variable n as the array size

    printf ("\n");
    printf ("\n");

    for (int i = 0; i < n; i++){
        printf ("Enter a number for the array: ");
        scanf ("%d", &A[i]);   //input elements to the array
    }

    printf ("\n");
    printf ("------------------------------------------------------------------");

    printf ("\n");

    printf ("\nNumber of ordered pairs = %d \n", MergeSort(A, n));   //calling MergeSort function & display the number of pairs
    printf ("\n");
    printf ("------------------------------------------------------------------");
    printf ("\n");

        return 0;
}
-- INSERT --                                                    113,2         Bot
```

In the main function an array is created with n elements and n can be any number.

In here as the array 1, 4, 3, 2 and 5 have been entered. So, the number of ordered pairs should be 3 and the ordered pairs are printed on the terminal.

```
kali@kali: ~                                                    12:53:38  98%
File  Actions  Edit  View  Help
kali@kali:~$ gcc IT19001180_FA2.c -o b
kali@kali:~$ ./b
------------------------------------------------------------------

Enter a number for n: 5

------------------------------------------------------------------

Enter a number for the array: 1
Enter a number for the array: 4
Enter a number for the array: 3
Enter a number for the array: 2
Enter a number for the array: 5

------------------------------------------------------------------

(4,3)
(3,2)
(4,2)
Number of ordered pairs = 3

------------------------------------------------------------------
kali@kali:~$
```