LAB SUBMISSION – 1

Meher Shrishti Nigam – 20BRS1193

1- Consider a situation where a class 'Graduation' has 10 students. Exam has been conducted and all students got their marks. Take this class Graduation[] as an array and implement the following

- **insert students' marks. At the same time suppose you forget one student marks and now you need to insert the marks somewhere in the middle of the list (you need to implement all the cases of insertion).**
- **display the marks obtained by each student. (You can assume the students marks randomly).**

3- Consider the Graduation Problem as in Q-1. So now authority have suggested to give students grace of 5 marks (but to only those who are below 70 marks). Improve the program with the given modification. Also try to search whether any student got the marks exactly 90.

Solution in C

```c
#include <stdio.h>
void Insert(int arr[], int n, int num, int k)
{
    // If insertion at end
    if(k == n + 1)
    {
        arr[n] = num;
        return;
    }
    // if insertion at any other position (not end)
    int j = n;
    while(j >= (k - 1))
    {
        arr[j] = arr[j - 1];
        j--;
    }
    arr[k - 1] = num;
    return;
}



typedef struct Exactly_90{
    int count;
    int positions[10];
}Exactly_90;

void Modify_Under_70(int arr[], int n)
```

```
{
    for(int i = 0; i < n; i++)
    {
        if(arr[i] < 70)
        {
            arr[i] += 5;
        }
    }
}

Exactly_90 Exactly_90_Counter(int arr[], int n)
{
    Exactly_90 answer;
    answer.count = 0;
    for(int i = 0; i < n; i++)
    {
        if(arr[i] == 90)
        {
            answer.positions[answer.count] = i;
            answer.count++;
        }
    }
    return answer;
}
void display(int arr[], int n)
{
    for(int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main()
{
    int graduation[10] = {67, 45, 93, 90, 68, 83, 90, 65, 90};
    // Marks before Insertion
    printf("Marks before Insertion of missing value - \n");
    display(graduation, 9);

    // lets say we forgot to insert marks m at position pos
    int m, pos;
```

```c
    printf("Enter missing marks: \n");
    scanf("%d", &m);
    printf("Enter missing marks's position: \n");
    scanf("%d", &pos);

    Insert(graduation, 9, m, pos);

    // Marks after Insertion
    printf("Marks after Insertion of missing value - \n");
    display(graduation, 10);

    // Improve Marks
    Modify_Under_70(graduation, 10);
    Exactly_90 ans = Exactly_90_Counter(graduation, 10);
    printf("Improved marks: \n");
    display(graduation, 10);
    printf("\n");

    // Count the number of 90's
    printf("Number of 90's: %d \nTheir positions: ", ans.count);
    for(int i = 0; i < ans.count; i++)
    {
        printf("%d ", ans.positions[i] + 1);
    }
}
```

```
PS C:\Users\meher\OneDrive\Documents\Sem 3\DSA\Lab\Lab_1> gcc Task_1_3.c
PS C:\Users\meher\OneDrive\Documents\Sem 3\DSA\Lab\Lab_1> ./a.exe
Marks before Insertion of missing value -
67 45 93 90 68 83 90 65 90
Enter missing marks:
34
Enter missing marks's position:
10
Marks after Insertion of missing value -
67 45 93 90 68 83 90 65 90 34
Improved marks:
72 50 93 90 73 83 90 70 90 39

Number of 90's: 3
Their positions: 4 7 9
PS C:\Users\meher\OneDrive\Documents\Sem 3\DSA\Lab\Lab_1>
```

Concepts:
To **insert a new element**, we first check if we are adding the element to the end of the array or at any other position, as adding an element to the end is easier (just add the element and increment count).

If its at any other position, we start shifting the elements of the array from the end to the right, so the 6th element becomes the 7th, the 5th becomes the 6th … till we reach the position at which we want to insert. We place the element in that space and increment the counter.

0	1	2	3	4	5	6
67	45	76	32	98	45	
Add 43 at 4th position						
67	45	76	32	98	45	45
67	45	76	32	98	98	45
67	45	76	32	32	98	45
67	45	76	43	32	98	45

Here, there are only 10 students, so we don't worry about reaching the end of the array.

Modification of marks (adding grace marks) is done by **traversing** the array and adding a specified value to each element.

To count the number of 90's and track their postions (i.e. 0 based positions), a struct is used. int **count stores the number of occurances** of 90 in the array, int positions[10] array tracks their positions.

Alternative Methods

Insertion can be done by creating a new array, by copying the left half of the array, inserting a new element, and copying the right half of the array. Space required will be more.

2- Consider a situation of a hospital where you have to list down all the important vaccine's cost given to the kids. Write a program to list the cost of the vaccine (at 0th index consider that as first vaccine)

- **Insert the price of each vaccine.**
- **Improve the above program to make some random insertion.**
- **Consider due to some pandemic situation the prices of each vaccine came down by 4 rs/-. Perform the reduction in price on each of the available list.**
- **Suppose there is one wrong entry (at say position 1st). Now you have to delete the particular entry from the list.**

4- Consider the hospital program in Q-2. Implement the copying operation on the array you have considered.

Solution in C

```c
#include <stdio.h>
#include <ctype.h>
int Insert(int arr[], int n)
{
    int reps;
    printf("Number of insertions: ");
    scanf("%d", &reps);
    for(int i = 0; i < reps; i++)
    {
        int x, k;
        printf("Number to insert: ");
        scanf("%d", &x);
        printf("Position to insert: ");
        scanf("%d", &k);
        if(k == n) // If insertion at end
        {
            arr[n - 1] = x;
            continue;
        }
        int j = n-1; // if insertion at any other position (not end)
        while(j >= (k - 1))
        {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[k - 1] = x;
        n++;
    }
    return reps;
}
void Modify(int arr[], int n)
{
    printf("Number to add to each element (Ex:Add -
4 to each element) :");
    int x;
    scanf("%d", &x);
    for(int i = 0; i < n; i++)
    {
        arr[i] += x;
    }
}
```

```c
int Delete(int arr[], int n)
{
    int reps;
    printf("Number of deletions: ");
    scanf("%d", &reps);
    for(int i = 0; i < reps; i++)
    {
        int k;
        printf("Position to delete (n = %d): ", n);
        scanf("%d", &k);
        // Deletion at end
        if(k == n - 1)
        {
            n--;
        }
        else // Deletion at any other position
        {
            int j = k - 1;
            while(j != n - 1)
            {
                arr[j] = arr[j + 1];
                j++;
            }
            n--;
        }
    }
    return reps;
}


void display(int arr[], int n)
{
    for(int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
    fflush(stdout);
}
```

```c
void Copy(int arr[], int ans[], int n)
{
    for(int i = 0; i < n; i++)
    {
        ans[i] = arr[i];
    }
}

int main()
{
    int vaccines[100];
    int n;

    printf("Intialize the prices of vaccines\n");
    printf("Number of vaccines: ");
    scanf("%d", &n);

    for(int i = 0; i < n; i++)
    {
        printf("Price of vaccine number %d : ", i+1);
        scanf("%d", &vaccines[i]);
    }
    int vaccines_copy[100];
    char choose = 'X';

    do
    {
        printf("I - Randomly Insert a price of vaccine\n");
        printf("M - Price modification\n");
        printf("D - Delete entry\n");
        printf("X - Exit menu\n");
        printf("C - Copy array\n");

        fflush(stdout);
        scanf(" %c", &choose);

        if(toupper(choose) == 'I')
        {
            n += Insert(vaccines, n);
            display(vaccines, n);
        }
```

```
        else if(toupper(choose) == 'M')
        {
            Modify(vaccines, n);
            display(vaccines, n);
        }
        else if(toupper(choose) == 'D')
        {
            n -= Delete(vaccines, n);
            display(vaccines, n);
        }
        else if(toupper(choose) == 'C')
        {
            Copy(vaccines, vaccines_copy, n);
            display(vaccines_copy, n);
        }
        else("Invalid response");
    }while(choose != 'X');
}
```

Concepts:

(In C, during the declaration of an array only a constant value has to be used. Here I have assumed that the number of vaccines will not exceed 100. If we coded this in C++ then we could have declared the array with n terms, or used vectors to extend the array as we like.)

The array containing the prices of vaccines is initialized at first. A menu option is provided to insert, modify, delete or copy the vaccines array.

Insertion is done by **shifting to right**, as explained in the problem above.

Deletion is done by taking at which the value has to be deleted. If the position is at the end, then just the array length is decremented. If deletion is to be done at any other position, then the whole array to the right of that postion is **shifted one position to the left**.

0	1	2	3	4	5	6
67	45	76	43	32	98	45
Delete element at 4th position						
67	45	76	32	32	98	45
67	45	76	32	98	98	45
67	45	76	32	98	45	45
67	45	76	32	98	45	x

Modification of prices is done by **traversing** the array and adding a specified value to each element.

Copy of array is also made by **traversing** the whole original array and copying (assigning) each value to the corresponding element of the new array.

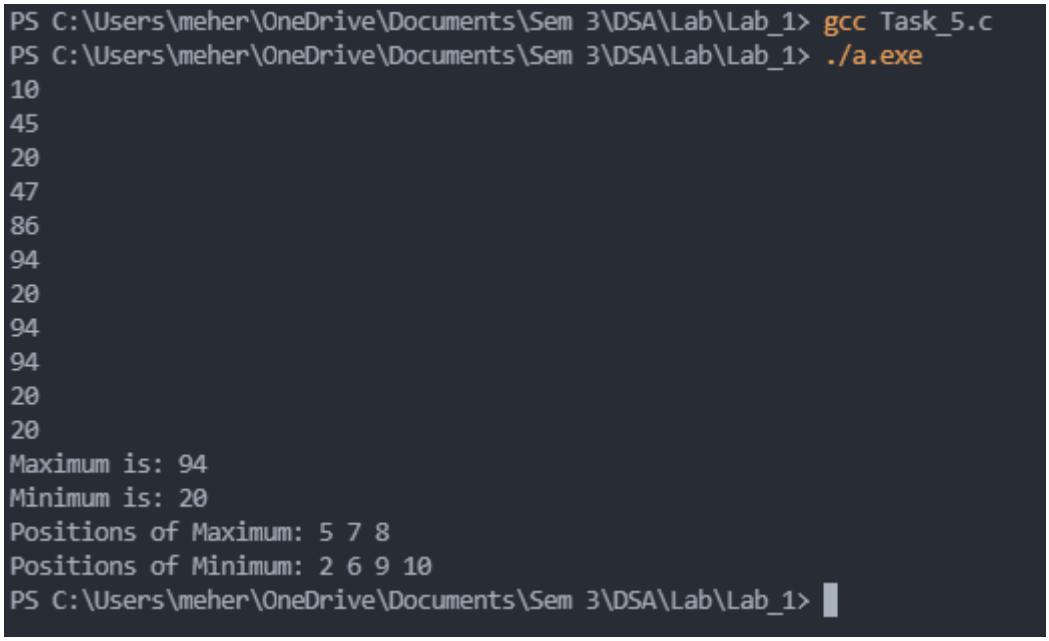
Alternative Methods

Insertion and deletion can be done by creating a new array, by copying the left half of the array, inserting a new element or skipping an element to delete, and copying the right half of the array. Space required will be more.

5- Finding Max and Min Element in the Array using Brute Force Approach.

```
#include <stdio.h>

typedef struct max_min{
    int max;
    int min;
    int c_max;
    int c_min;
    int pos_max[100];
    int pos_min[100];
}max_min;

max_min Max_Min_Element(int arr[], int n)
{
    max_min arr_max_min;
    arr_max_min.c_max = arr_max_min.c_min = 1;
    arr_max_min.max = arr_max_min.min = arr[0];
    arr_max_min.pos_max[0] = arr_max_min.pos_min[0] = 0;
    for(int i = 0; i < n; i++)
    {
        if(arr[i] > arr_max_min.max)
        {
            arr_max_min.pos_max[0] = i;
            arr_max_min.c_max = 1;
            arr_max_min.max = arr[i];
        }
        else if(arr[i] == arr_max_min.max)
        {
            arr_max_min.pos_max[arr_max_min.c_max] = i;
            arr_max_min.c_max++;
        }
        else if(arr[i] < arr_max_min.min)
        {
            arr_max_min.pos_min[0] = i;
            arr_max_min.c_min = 1;
            arr_max_min.min = arr[i];
        }
```

```
        else if(arr[i] == arr_max_min.min)
        {
            arr_max_min.pos_min[arr_max_min.c_min] = i;
            arr_max_min.c_min++;
        }
    }
    return arr_max_min;
}


void display (max_min m)
{
    printf("Maximum is: %d\n", m.max);
    printf("Minimum is: %d\n", m.min);

    printf("Positions of Maximum: ");
    for(int i = 0; i < m.c_max; i++)
    {
        printf("%d ", m.pos_max[i] + 1);
    }
    printf("\nPositions of Minimum: ");
    for(int i = 0; i < m.c_min; i++)
    {
        printf("%d ", m.pos_min[i] + 1);
    }
}
int main()
{
    int n;
    scanf("%d", &n);
    int arr[100];
    for(int i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    max_min answer = Max_Min_Element(arr, n);
    display(answer);
}
```



Concepts

To find the max/min of element in an array we initialize the max/min to the first array element. Then we traverse the array, if there is an element greater/lesser that max/min we change max/min accordingly.

Here an array is used to track all the positions of the max/min in an array. If during traversal we come across an element equal to the current max/min, we store their position (0 based) in an array and increment the count of max/min. If we come across a new max/min, we set the count of max/min to 1 and store the new postions in the array.

A struct is used to return all this information in one loop.

Alternative Methods

Sort the array. The starting element is the minimum and the last element is maximum.
Divide and Conquer method.