Q. Insert and element at the end of the array and at a given position, Delete and element from the end of the array and delete an element from a given position in an array.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// array io functions
int *input_array(int *size, char *message);
void print_array(int *array, int size);

// insertion functions
void append_elmnt(int *array, int *size, int element);
int insert_elmnt(int *array, int *size, int element, int index);

// deletion functions
int pop_elmnt(int *arr, int *size);
int remove_elmnt(int *arr, int *size, int index);

int main()
{
    int *array, size = 0, choice, element, position = -1, flag;
    while (1)
    {
        // printing the menu
        printf("\n");
        printf("1.Element Insertion at the end\n2.Element Insertion at an Positon\n3.Pop
an Element\n4.Delete an Element by Postion\n5.Exit\n\n");
        printf("Enter your choice:\n--> ");

        scanf("%d", &choice);
        printf("\n");

        switch (choice)
        {
            {
            case 1:
                array = input_array(&size, "Enter the elements of the array:\n");

                // reading the element to be inserted from the user
                printf("\nEnter the element to insert:\n--> ");
                scanf("%d", &element);
                printf("\n");

                // inserting the element at the end of the array
                append_elmnt(array, &size, element);
                printf("Modified array is:\n");
                print_array(array, size);
                break;
```

```
            case 2:
                array = input_array(&size, "Enter the elements of the array:\n");

                // reading the element to be inserted from the user
                printf("\nEnter the element to insert:\n--> ");
                scanf("%d", &element);
                printf("\nEnter the position at which you want to insert the element:\n-->
");
                scanf("%d", &position);
                printf("\n");

                // inserting the element at the given index in the arrya
                flag = insert_elmnt(array, &size, element, position - 1);
                if(!flag){
                    printf("Modified array is:\n");
                    print_array(array, size);
                }
                else{
                    printf("Enter a valid position!\n");
                }

                break;

            case 3:
                array = input_array(&size, "Enter the elements of the array:\n");

                // removing the element from the end
                flag = pop_elmnt(array, &size);
                if(!flag){
                    printf("Modified array is:\n");
                    print_array(array, size);
                }
                else{
                    printf("Enter a valid position!\n");
                }
                break;

            case 4:
                array = input_array(&size, "Enter the elements of the array:\n");

                // reading the element to be inserted from the user
                printf("\nEnter the position of element you want to remove:\n--> ");
                scanf("%d", &position);
                printf("\n");

                // removing the element from the postion
                flag = remove_elmnt(array, &size, position - 1);
                if(!flag){
                    printf("Modified array is:\n");
                    print_array(array, size);
```

```
                }
                else{
                    printf("Enter a valid position!\n");
                }
                break;

            case 5:
                return 0;

            default:
                printf("Wrong choice!!!\n");
                break;
            }
        }
    }
}

// array io functions
int *input_array(int *size, char *message)
{
    // initializing variables
    int *array;
    // reading the size of the array from the user
    printf("Enter the size of the array:\n--> ");
    scanf("%d", size);

    // initializing the array of size 'size'
    array = (int *)malloc(*size * sizeof(int)); // using *size to access the size(value)
of the array

    // reading the elements of the array from the user
    if(*size > 0) {
        printf("\n%s", message);
        for (int i = 0; i < *size; i++)
        {
            printf("array[%d]: ", i);
            scanf("%d", &array[i]);
        }
    }

    return array;
}

void print_array(int *array, int size)
{
    printf("[");
    for (int i = 0; i < size; i++)
    {
        printf("%d ", array[i]);
    }
    printf("]\n");
```

```c
}

// insertion functions
void append_elmnt(int *array, int *size, int element)
{
    array = (int *)realloc(array, *size * sizeof(int) + 1);
    *size = *size + 1;
    array[*size - 1] = element;
}

int insert_elmnt(int *array, int *size, int element, int index)
{
    if (index > *size)
    {
        return -1;
    }

    // increasing the size of the array
    array = (int *)realloc(array, *size * sizeof(int) + 1);
    *size = *size + 1;

    // shifting the elements of the array by one postion(1)
    for (int i = index; i < *size; i++)
    {
        array[i + 1] = array[i];
    }

    array[index] = element;

    return 0;
}

// deletion functions
int pop_elmnt(int *array, int *size)
{
    if (*size == 0)
    {
        return -1;
    }
    // decreasing size of the array by one postion
    array = (int *)realloc(array, *size * sizeof(int) - 1);
    *size = *size - 1;

    return 0;
}

int remove_elmnt(int *array, int *size, int index)
{
    if (index > *size)
    {
        return -1;
```

```
    }

    // shifting the elements by one postion (-1)
    for (int i = index; i < *size; i++)
    {
        array[i] = array[i+1];
    }

    // decreasing size of the array by one postion
    array = (int *)realloc(array, *size * sizeof(int) - 1);
    *size = *size - 1;

    return 0;
}
```

Q. Perform Linear Search, Binary Search and Bubble sort on an array.

```c
// Menu driven program to search an element in an array using linear search and binary search and bubble sort

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// array manipulation functions
int *input_array(int *size, char *message);
void print_array(int *array, int size);

// search functions
int linear_search(int *arr, int element, int size);
int binary_search(int *arr, int element, int size);

// sorting functions
void bubble_sort(int *arr, int size);

int main()
{
    int *array, size = 0, choice, element, postion = -1;
    while (1)
    {
        // printing the menu
        printf("\n");
        printf("1.Linear Search\n2.Binary Search\n3.Bubble Sort\n4.Exit\n\n");
        printf("Enter your choice:\n--> ");

        scanf("%d", &choice);
        printf("\n");

        switch (choice)
        {
            {
            case 1:
                array = input_array(&size, "Enter the elements of the array:\n");

                // reading the element to be searched from the user
                printf("\nEnter the element to search:\n--> ");
                scanf("%d", &element);
                printf("\n");

                // searching the element using linear search
                postion = linear_search(array, element, size);
                printf("Element '%d' is found at postion: %d\n", element, postion + 1);
                break;
            case 2:
```

```c
                array = input_array(&size, "Enter the element in sorted manner:\n");

                // reading the element to be searched from the user
                printf("\nEnter the element to search:\n--> ");
                scanf("%d", &element);
                printf("\n");

                // searching the element using binary search
                postion = binary_search(array, element, size);
                printf("Element '%d' is found at postion: %d\n", element, postion + 1);
                break;

            case 3:
                array = input_array(&size, "Enter the elements of the array:\n");

                // sorting the array using bubble sort
                bubble_sort(array, size);
                printf("\nThe sorted array is:\n");
                print_array(array, size);
                break;

            case 4:
                return 0;

            default:
                printf("Wrong choice!!!\n");
                break;
            }
        }
    }
}

// array manipulation functions
int *input_array(int *size, char *message)
{
    // initializing variables
    int *array;
    // reading the size of the array from the user
    printf("Enter the size of the array:\n--> ");
    scanf("%d", size);

    // initializing the array of size 'size'
    array = (int *)malloc(*size * sizeof(int)); // using *size to access the size(value)
of the array

    // reading the elements of the array from the user
    if(*size > 0) {
        printf("\n%s", message);
        for (int i = 0; i < *size; i++)
        {
            printf("array[%d]: ", i);
```

```
            scanf("%d", &array[i]);
        }
    }

    return array;
}

void print_array(int *array, int size)
{
    printf("[");
    for (int i = 0; i < size; i++)
    {
        printf("%d ", array[i]);
    }
    printf("]\n");
}

// search functions
int linear_search(int *array, int element, int size)
{
    for (int i = 0; i < size; i++)
    {
        if (array[i] == element)
        {
            return i;
        }
    }
    return -1;
}

int binary_search(int *array, int element, int size)
{
    int lowerBound = 0, upperBound = size - 1, mid;
    for (int i = 0; i < size; i++)
    {
        mid = (lowerBound + upperBound) / 2;
        if (array[mid] == element)
        {
            return mid;
        }
        else if (array[mid] > element)
        {
            upperBound = mid - 1;
        }
        else
        {
            lowerBound = mid + 1;
        }
    }
}
```

```c
// sorting functions
void bubble_sort(int *array, int size)
{
    for (int i = 0; i < size; i++)
    {
        for (int j = 0; j < size; j++)
        {
            if (array[i] < array[j])
            {
                int temp = array[i];
                array[i] = array[j];
                array[j] = temp;
            }
        }
    }
}
```