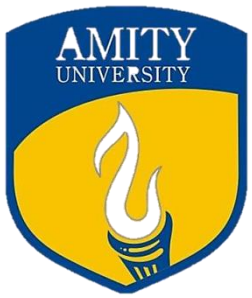
Data Structures using C

Lab File

Submitted to:

AMITY UNIVERSITY UTTTAR PARDESH



In partial fulfilment of the requirements for the award of the degree of Bachelor of technology

In

Computer Science & Engineering

By

Faizan Ahmed Syed
A23055223216
CSE 4-X Semester 3

Submitted to:

Dr. Smriti Sehgal

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY
AMITY UNIVERSITY UTTAR PARDESH
NOIDA (U.P.)

LIST OF EXPERIMENTS

S.No	Experiment	Date	Page No.	Sign

S.No	Experiment	Date	Page No.	Sign

S.No	Experiment	Date	Page No.	Sign

Lab 1

Program:

Write a menu driven program for the insertion of elements in the beginning, middle, and the end of a 1-D array.

Source Code:

```
#include<stdio.h>
void beginning(int new_element, int n, int arr[]) {
    for(int i = n; i > 0; i--) {
        arr[i] = arr[i - 1];
    }
    arr[0] = new_element;
    printf("New array: ");
    for(int i = 0; i <= n; i++){
        printf("%d ", arr[i]);
    } printf("\n\n");
}
void middle(int new_element, int arr[], int position, int n) {
    for(int i = n; i >= position; i--) {
        arr[i] = arr[i - 1];
        }
        arr[position] = new_element;
    printf("New array: ");
    for(int i = 0; i <= n; i++)
        printf("%d ", arr[i]);
     printf("\n\n");
}
void end(int new_element, int n, int arr[]) {
    arr[n] = new_element;
    printf("New array: ");
    for(int i = 0; i <= n; i++){
        printf("%d ", arr[i]);
    } printf("\n\n");
}
int main() {
    int arr[100];
    int n;
    int option;
    int new_element;
    int position;
    char repeat;

    printf("\n -> Insertion Option Menu:\n");
    printf("\nSelect the number for the task you want to perform.\n");
    printf("1- Beginning Insertion.\n2- Middle Insertion.\n3- End
Insertion.\n");
```

```
printf("\nCreate an array: \n");
printf("Enter number of elements: ");
scanf("%d", &n);
printf("\nEnter the %d elements of the array: \n", n);
for(int i = 0; i < n; i++){
    printf("Element %d: ", i);
    scanf("%d", &arr[i]);
}
printf("The array is: \n");
for(int i = 0; i < n; i++) {
    printf("%d ", arr[i]);
} printf("\n\n");

do {
    printf("Select the task (1-3): ");
    scanf("%d", &option);
    printf("\n");

    switch(option) {
            case 1:
                printf("Enter value to be inserted: ");
                scanf("%d", &new_element);
                beginning(new_element, n, arr);
                n = n + 1;
                break;
            case 2:
                printf("Enter value to be inserted: ");
                scanf("%d", &new_element);
                printf("Which position do you want to insert at: ");
                scanf("%d", &position);
                if(position < 0 || position > n) {
                    printf("Invalid position...\n");
                    n = n + 1;
                    break;
                }
                middle(new_element, arr, position, n);
                break;
            case 3:
                printf("Enter value to be inserted: ");
                scanf("%d", &new_element);
                n = n + 1;
                end(new_element, n, arr);
                break;
            default:
                printf("Selected option does not exist.\n");
                printf("Please choose from 1-3 only...\n\n");
                break;
        }
```

```
            printf("Do you want to continue (Y/N): ");
            scanf(" %c", &repeat);
            printf("\n");
    }
    while(repeat == 'Y' || repeat == 'y');

    return 0;
}
```

Output:

```
 -> Insertion Option Menu:

Select the number for the task you want to perform.
1- Beginning Insertion.
2- Middle Insertion.
3- End Insertion.

Create an array:
Enter number of elements: 5

Enter the 5 elements of the array:
Element 0: 2
Element 1: 4
Element 2: 6
Element 3: 8
Element 4: 10
The array is:
2 4 6 8 10

Select the task (1-3): 1

Enter value to be inserted: 1
New array: 1 2 4 6 8 10

Do you want to continue (Y/N): y

Select the task (1-3): 2

Enter value to be inserted: 5
Which position do you want to insert at: 3
New array: 1 2 4 5 6 8 10

Do you want to continue (Y/N): y

Select the task (1-3): 3

Enter value to be inserted: 11
New array: 1 2 4 5 6 8 10 11

Do you want to continue (Y/N): n

PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C> ^C
```

Lab 2

Program:

Write a menu driven program for the deletion of elements from the beginning middle and end of a 1-D array.

Source Code:

```
#include<stdio.h>
 void beginning(int n, int arr[]) {
    for(int i = 0; i < n - 1; i++) {
        arr[i] = arr[i + 1];
    }
    n = n - 1;
    printf("New array: ");
    for(int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    } printf("\n\n");
}
void middle(int arr[], int position, int n) {
    for(int i = position; i <= n - 1; i++) {
        arr[i] = arr[i + 1];
        }
        n = n - 1;
    printf("New array: ");
    for(int i = 0; i < n; i++)
        printf("%d ", arr[i]);
     printf("\n\n");
}
void end(int n, int arr[]) {
    n = n - 1;
    printf("New array: ");
    for(int i = 0; i < n; i++){
        printf("%d ", arr[i]);
    } printf("\n\n");
}
 int main() {
    int arr[100];
    int n;
    int option;
    int position;
    char repeat;

    printf("\n -> Deletion Option Menu:\n");
    printf("\nSelect the number for the task you want to perform.\n");
    printf("1- First Element Deletion.\n2- Middle Element Deletion.\n3- Last
Element Deletion.\n");
```

```
    printf("\nCreate an array: \n");
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("\nEnter the %d elements of the array: \n", n);
    for(int i = 0; i < n; i++){
        printf("Element %d: ", i);
        scanf("%d", &arr[i]);
    }
    printf("The array is: \n");
    for(int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    } printf("\n\n");

    do {
        printf("Select the task (1-3): ");
        scanf("%d", &option);
        printf("\n");

        switch(option) {
                case 1:
                    beginning(n, arr);
                    n = n - 1;
                    break;
                case 2:
                    printf("Which position element do you want to delete: ");
                    scanf("%d", &position);
                    if(position < 0 || position > n) {
                        printf("Invalid position...\n");
                        break;
                    }
                    middle(arr, position, n);
                    n = n - 1;
                    break;
                case 3:
                    end(n, arr);
                    break;
                default:
                    printf("Selected option does not exist.\n");
                    printf("Please choose from 1-3 only...\n\n");
                    break;
            }
            printf("Do you want to continue (Y/N): ");
            scanf(" %c", &repeat);
            printf("\n");
    }
    while(repeat == 'Y' || repeat == 'y');
    return 0;
}
```

Output:

```
 -> Deletion Option Menu:

Select the number for the task you want to perform.
1- First Element Deletion.
2- Middle Element Deletion.
3- Last Element Deletion.

Create an array:
Enter number of elements: 8

Enter the 8 elements of the array:
Element 0: 2
Element 1: 4
Element 2: 6
Element 3: 8
Element 4: 10
Element 5: 12
Element 6: 14
Element 7: 16
The array is:
2 4 6 8 10 12 14 16

Select the task (1-3): 1

New array: 4 6 8 10 12 14 16

Do you want to continue (Y/N): y

Select the task (1-3): 2

Which position element do you want to delete: 4
New array: 4 6 8 10 14 16

Do you want to continue (Y/N): y

Select the task (1-3): 3

New array: 4 6 8 10 14

Do you want to continue (Y/N): n

PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C>
```

Lab 3

Program:

Write a program to perform linear search and binary search on a 1-D array.

Source Code:

```c
#include<stdio.h>
int linear_search(int arr[], int n, int val) {
    for(int i = 0; i < n; i++) {
        if(arr[i] == val) {
            return i;
        }
    }
    return  -1;
}
int binary_search(int arr[], int low, int high, int val) {
    if(high >= low) {
        int mid = low + (high - low) / 2;
        if(arr[mid] == val)
            return mid;
        if(arr[mid] > val)
            return binary_search(arr, low, mid-1, val);

        return binary_search(arr, mid+1, high, val);
    }
    return -1;
}

int main() {
    int arr[100], n, val, index, optn, contn;

    printf("\nNote: array must be sorted...\n");
    printf("Create an array:\n");
    printf("\nEnter the number of elements (n) in the array: ");
    scanf("%d", &n);
    printf("\nEnter the %d elements of the array:\n", n);
    for(int i = 0; i < n; i++) {
        printf("Element %d: ", i);
        scanf("%d", &arr[i]);
    }
    printf("\nThe array created:\n");
    for(int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    } printf("\n\n");

    do {
        printf("Choose the search you want to perform:\n");
```

```
        printf(" 1- Linear search.\n 2- Binary search.\n");
        printf("Enter option: ");
        scanf("%d", &optn);

        switch(optn) {
            case 1:
                printf("\nLinear search:\n");
                printf("Enter the value you want to search: ");
                scanf("%d", &val);
                index = linear_search(arr, n, val);
                if(index == -1) {
                    printf("Element is not present in given array.\n");
                }
                else {
                    printf("The value %d is present in array at index: %d\n",
val, index);
                }
                break;
            case 2:
                printf("\nBinary search:\n");
                printf("Enter the value you want to search: ");
                scanf("%d", &val);
                index = binary_search(arr, 0, n-1, val);
                if(index == -1) {
                    printf("Element is not present in given array.\n");
                }
                else
                    printf("The value %d is present in array at index: %d\n",
val, index);
                break;
            default:
                printf("Incorrect option...\nPlease choose from provided
menu.\n");
                break;
        }
        printf("\nDo you want to continue(Y/N): ");
        scanf(" %c", &contn);
    }
    while(contn == 'Y' || contn == 'y');

    printf("\nProgram terminated successfully...\n\n");

    return 0;
}
```

Output:

```
Note: array must be sorted...
Create an array:

Enter the number of elements (n) in the array: 5

Enter the 5 elements of the array:
Element 0: 2
Element 1: 4
Element 2: 7
Element 3: 10
Element 4: 15

The array created:
2 4 7 10 15

Choose the search you want to perform:
 1- Linear search.
 2- Binary search.
Enter option: 1

Linear search:
Enter the value you want to search: 7
The value 7 is present in array at index: 2

Do you want to continue(Y/N): y
Choose the search you want to perform:
 1- Linear search.
 2- Binary search.
Enter option: 2

Binary search:
Enter the value you want to search: 10
The value 10 is present in array at index: 3

Do you want to continue(Y/N): n

Program terminated successfully...

PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C>
```

Lab 4

Program:

Write a menu driven program to perform operations on 2-D arrays. The program must perform the operations: addition, subtraction, multiplication, transpose, diagonal elements sum, print upper triangle, and print lower triangle.

Source Code:

```
#include<stdio.h>
#define MAX 50
void array(int num, int n, int m, int a[MAX][MAX]) {
    printf("Enter the elements of the %d array:\n", num);
    for(int i = 0;i < n; i++) {
        printf("Row %d of array:\n", i);
        for(int j = 0; j < m; j++) {
            printf("Element %d: ", j);
            scanf("%d", &a[i][j]);
        }
    }
    printf("\nArray %d created:\n", num);
    for(int i = 0; i < n; i++) {
        printf("   ");
        for(int j = 0; j < m; j++) {
            printf("%d ", a[i][j]);
        }
        printf("\n");
    } printf("\n");
}
void addition(int n, int m, int a1[MAX][MAX], int a2[MAX][MAX]) {
    int a_result[MAX][MAX];
    for(int i = 0; i < n; i++) {
        printf(" ");
        for(int j = 0; j < m; j++) {
            a_result[i][j] = a1[i][j] + a2[i][j];
            printf("%d ", a_result[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}
void subtraction(int n, int m, int a1[MAX][MAX], int a2[MAX][MAX]) {
    int a_result[MAX][MAX];
    for(int i = 0; i < n; i++) {
        printf(" ");
        for(int j = 0; j < m; j++) {
            a_result[i][j] = a1[i][j] - a2[i][j];
```

```
            printf("%d ", a_result[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}
void multiplication(int n, int m, int a1[MAX][MAX], int a2[MAX][MAX]) {
    int a_result[MAX][MAX];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            a_result[i][j] = 0;
            for (int k = 0; k < n; k++) {
                a_result[i][j] += a1[i][k] * a2[k][j];
            }
            printf("%d\t", a_result[i][j]);
        }
        printf("\n");
    }
}
void transpose(int n, int m, int a[MAX][MAX]) {
    int a_result[MAX][MAX];
    for(int i = 0; i < n; i++) {
        for(int j = 0; j < m; j++) {
            a_result[i][j] = a[j][i];
            printf("%d ", a_result[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}
void diagonal_Sum(int n, int m, int a[MAX][MAX], int optn) {
    int sum = 0;
    if(optn == 1) {
        for(int i = 0; i < n ; i++)
            for(int j = 0; j < m; j++) {
                if(i == j) {
                    sum += a[i][j];
                }
            }
    }
    Else {
        for(int i = 0; i < n; i++)
            for(int j = 0; j < m; j++) {
                int c = i + j;
                if(c == (n-1)) {
                    sum += a[i][j];
                }
            }
```

```
    }
    printf("Sum of diagonal is: %d\n", sum);
}
void upper_Triangle(int n, int m, int a[MAX][MAX]) {
    for(int i = 0; i < n; i++) {
        for(int j = 0; j < m; j++) {
            printf("%d ", a[i][j]);
        }
        m--;
        printf("\n");
    }
    printf("\n");
}
void lower_Triangle(int n, int m, int a[MAX][MAX]) {
    for(int i = 0; i < n; i++) {
        for(int j = 0; j < m; j++) {
            int c = i + j;
            if(c > (n-1)) {
                printf("%d ", a[i][j]);
            }
            else
                printf("  ");
        }
        printf("\n");
    }
    printf("\n");
}
int main() {
    int option, arrChoice;
    int n, m, num, transpose_of, up_tri, lo_tri, optn;
    int a1[MAX][MAX], a2[MAX][MAX];
    char exit;

    //Creating 2 2D arrays from user input:
    printf("Create array 1:\n");
    printf("Enter number of rows of array: ");
    scanf("%d", &n);
    printf("Enter number of columns of array: ");
    scanf("%d", &m);
    array(1, n, m, a1);

    printf("Create array 2:\n");
    array(2, n, m, a2);

    //Printing menu and taking option input:
    printf("\n****** 2D Array Operations ******\n\n");
    printf("Choose the operation you want to perform:\n");
```

```
    printf(" 0- Create new arrays.\n 1- Addition.\n 2- Subtraction.\n 3-
Multiplication.\n 4- Transpose.\n 5- Diagonal sum.\n 6- Upper triangle.\n 7-
Lower triangle.");
    do {
        printf("\nSelect operation: ");
        scanf("%d", &option);
        printf("\n");

        switch(option) {
            case 0:
                printf("Create array 1:\n");
                printf("Enter number of rows of array: ");
                scanf("%d", &n);
                printf("Enter number of columns of array: ");
                scanf("%d", &m);
                array(1, n, m, a1);
                printf("Create array 2:\n");
                array(2, n, m, a1);
                break;
            case 1:
                addition(n, m, a1, a2);
                break;
            case 2:
                subtraction(n, m, a1, a2);
                break;
            case 3:
                multiplication(n, m, a1, a2);
                break;
            case 4:
                printf("Choose array to transpose(1/2): ");
                scanf("%d", &transpose_of);
                if(transpose_of == 1) {
                    transpose(n, m, a1);
                }
                else {
                    transpose(n, m, a2);
                }
                break;
            case 5:
                printf("Choose diagonal:\n");
                printf(" 1- Decreasing diagonal.\n 2- Increasing
diagonal.\n");
                printf("Enter option: ");
                scanf("%d", &optn);
                printf("Choose array to sum the diagonal of(1/2): ");
                scanf("%d", &arrChoice);
                if(arrChoice == 1) {
                    diagonal_Sum(n, m, a1, optn);
```

```
                }
                else {
                    diagonal_Sum(n, m, a2, optn);
                }
                break;
            case 6:
                printf("Choose array to print upper triangle of(1/2): ");
                scanf("%d", &up_tri);
                if(up_tri == 1) {
                    upper_Triangle(n, m, a1);
                }
                else {
                    upper_Triangle(n, m, a2);
                }
                break;
            case 7:
                printf("Choose array to print lower triangle of(1/2): ");
                scanf("%d", &lo_tri);
                if(lo_tri == 1) {
                    lower_Triangle(n, m, a1);
                }
                else {
                    lower_Triangle(n, m, a2);
                }
                break;
            default:
                printf("Wrong option...\n");
                printf("Choose from given options (1 - 7).\n\n");
                break;
        }
        printf("Do you want to continue (Y/N): ");
        scanf(" %c", &exit);
    }
    while(exit == 'Y' || exit == 'y');
    printf("\nProgram terminated successfully...\n\n");
    return 0;
}
```

Output:

```
Create array 1:
Enter number of rows of array: 3
Enter number of columns of array: 3
Enter the elements of the 1 array:
Row 0 of array:
Element 0: 1
Element 1: 2
Element 2: 3
Row 1 of array:
Element 0: 4
Element 1: 5
Element 2: 6
Row 2 of array:
Element 0: 7
Element 1: 8
Element 2: 9

Array 1 created:
   1 2 3
   4 5 6
   7 8 9

Create array 2:
Enter the elements of the 2 array:
Row 0 of array:
Element 0: 9
Element 1: 8
Element 2: 7
Row 1 of array:
Element 0: 6
Element 1: 5
Element 2: 4
Row 2 of array:
Element 0: 3
Element 1: 2
Element 2: 1

Array 2 created:
   9 8 7
   6 5 4
   3 2 1
```

```
****** 2D Array Operations ******

Choose the operation you want to perform:
 0- Create new arrays.
 1- Addition.
 2- Subtraction.
 3- Multiplication.
 4- Transpose.
 5- Diagonal sum.
 6- Upper triangle.
 7- Lower triangle.
Select operation: 1

 10 10 10
 10 10 10
 10 10 10

Do you want to continue (Y/N): y

Select operation: 2

 -8 -6 -4
 -2 0 2
 4 6 8

Do you want to continue (Y/N): y

Select operation: 3

30      24      18
84      69      54
138     114     90
Do you want to continue (Y/N): y

Select operation: 4

Choose array to transpose(1/2): 1
1 4 7
2 5 8
3 6 9

Do you want to continue (Y/N): y

Select operation: 4

Choose array to transpose(1/2): 2
9 6 3
8 5 2
7 4 1
```

```
Do you want to continue (Y/N): y

Select operation: 5

Choose diagonal:
 1- Decreasing diagonal.
 2- Increasing diagonal.
Enter option: 1
Choose array to sum the diagonal of(1/2): 1
Sum of diagonal is: 15

Do you want to continue (Y/N): y

Select operation: 6

Choose array to print upper triangle of(1/2): 2
9 8 7
6 5
3

Do you want to continue (Y/N): y

Select operation: 7

Choose array to print lower triangle of(1/2): 1

    6
  8 9

Do you want to continue (Y/N): n

Program terminated successfully...

PS C:\Users\Faizan\Desktop\UNI STUFF\Semester 3\Data Structures in C>
```

Lab 5

Program:

Write a program to pass an array as a parameter to a function.

Source Code:

```
#include<stdio.h>
#define MAX 50
int array_multiple(int n, int m, int x, int arr[MAX][MAX]) {
    int new_arr[MAX][MAX];
    for(int i = 0; i < n; i++) {
        for(int j = 0; j < m; j++) {
            new_arr[i][j] = arr[i][j] * x;
            printf("%d ", new_arr[i][j]); }
        printf("\n");
    }
}
int main() {
    int n, m, x;
    int arr[MAX][MAX];

    printf("Create a 2-D array:\n");
    printf("Enter number of rows: ");
    scanf("%d", &n);
    printf("Enter number of columns: ");
    scanf("%d", &m);
    printf("Enter the elements of the array:\n");
    for(int i = 0;i < n; i++) {
        printf("Row %d of array:\n", i);
        for(int j = 0; j < m; j++) {
            printf("Element %d: ", j);
            scanf("%d", &arr[i][j]);  }
    }
    printf("\nArray created:\n");
    for(int i = 0; i < n; i++) {
        printf("   ");
        for(int j = 0; j < m; j++){
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    } printf("\n");
    printf("Enter the multiple value: ");
    scanf("%d", &x);
    array_multiple(n, m, x, arr);

    return 0;
}
```

Output:

```
 } ; if ($?) { .\Array_parameter }
Create a 2-D array:
Enter number of rows: 2
Enter number of columns: 3
Enter the elements of the array:
Row 0 of array:
Element 0: 1
Element 1: 2
Element 2: 3
Row 1 of array:
Element 0: 4
Element 1: 5
Element 2: 6

Array created:
   1 2 3
   4 5 6

Enter the multiple value: 3
3 6 9
12 15 18
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C>
```

Lab 6

Program:

Write a menu driven program to implement the stack operations of PUSH, POP, and Display.

Source Code:

```c
#include<stdio.h>
#define MAX 5
int stack[MAX];
int top = -1;
void PUSH(int val) {
    printf("\nEnter value to push to stack: ");
    scanf("%d", &val);
    if(top == (MAX - 1)) {
        printf("\nStack is full.\nCannot PUSH new value to stack.\n");
    }
    else {
        top++;
        stack[top] = val;
        printf("%d pushed to stack.\n", val);
    }
}
void POP() {
    if(top == -1) {
        printf("Stack is empty.\n");
    }
    else {
        printf("%d popped from stack.\n", stack[top]);
        top--;
    }
}
void display() {
    if(top == -1) {
        printf("Stack is empty.\nNo Display...\n");
    }
    else {
        printf("Stack Display:\n");
        for(int i = top; i >= 0; i--) {
            printf("    %d\n", stack[i]);
        }
    }
}
int main() {
    int optn, val;
    char contn;
    printf("Stack operations:\n 1- PUSH.\n 2- POP.\n 3- Display stack.\n");
```

```
    do {
        printf("Enter option (1-3): ");
        scanf("%d", &optn);
        switch(optn) {
            case 1:
                PUSH(val);
                break;
            case 2:
                POP();
                break;
            case 3:
                display();
                break;
            default:
                printf("Incorrect option.\nPlease choose from given
options.\n");
                break;
        }
        printf("\nDo you want to continue (Y/N): ");
        scanf(" %c", &contn);
    }
    while(contn == 'Y' || contn == 'y');
    printf("\nProgram terminates successfully...\n\n");
    return 0;
}
```

Output:

```
s } ; if ($?) { .\Stack_Operations }
Stack operations:
 1- PUSH.
 2- POP.
 3- Display stack.
Enter option (1-4): 1

Enter value to push to stack: 10
10 pushed to stack.

Do you want to continue (Y/N): y
Enter option (1-4): 1

Enter value to push to stack: 20
20 pushed to stack.

Do you want to continue (Y/N): y
Enter option (1-4): 1

Enter value to push to stack: 30
30 pushed to stack.

Do you want to continue (Y/N): y
Enter option (1-4): 3
Stack Display:
    30
    20
    10

Do you want to continue (Y/N): y
Enter option (1-4): 2
30 popped from stack.

Do you want to continue (Y/N): n

Program terminates successfully...

PS C:\Users\Faizan\Desktop\UNI STUFF\Semester 3\Data Structures in C>
```

Lab 7

Program:

Write a menu driven program to implement the queue operations of enqueue, dequeue, and display.

Source Code:

```
#include<stdio.h>
#define MAX 10
int Q[MAX];
int F = -1, R = -1;
void enqueue(int val) {
    if(R == (MAX - 1)) {
        printf("Queue is full.\n");
    }
    else if(F == -1 && R == -1) {
        F = 0;
        R = 0;
        printf("\nEnter value to add to queue: ");
        scanf("%d", &val);
        Q[R] = val;
        printf("%d added to queue.\n", val);
    }
    else {
        R++;
        printf("\nEnter value to add to queue: ");
        scanf("%d", &val);
        Q[R] = val;
        printf("%d added to queue.\n", val);
    }
}
void dequeue() {
    if(F == -1 && R == -1) {
        printf("Queue is already empty.\n");
    }
    else if(F == (MAX - 1) && R == (MAX - 1)) {
        F = -1;
        R = -1;
    }
    else {
        printf("%d deleted from queue.\n", Q[F]);
        F++;
    }
}
void display() {
    printf("\nDisplay Queue:\n");
    printf("  ");
```

```
    for(int i = F; i <= R; i++)
        printf("%d ", Q[i]);
    printf("\n");
}
int main() {
    int val, optn;
    char contn;
    printf("\nQueue operations:\n 1- Insert in queue.\n 2- Delete from
queue.\n 3- Display queue.\n");
    do {
        printf("Enter option (1-3): ");
        scanf("%d", &optn);
        switch(optn) {
            case 1:
                enqueue(val);
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            default:
                printf("Incorrect option.\nChoose from given options (1-
3).\n");
                break;
        }
        printf("\nDo you want to continue (Y/N): ");
        scanf(" %c", &contn);
    }
    while(contn == 'Y' || contn == 'y');
    printf("\nProgram terminated successfully...\n\n");
    return 0;
}
```

Output:

```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C> cd "c:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C>
ns } ; if ($?) { .\Queue_Operations }

Queue operations:
 1- Insert in queue.
 2- Delete from queue.
 3- Display queue.
Enter option (1-3): 1

Enter value to add to queue: 10
10 added to queue.

Do you want to continue (Y/N): y
Enter option (1-3): 1

Enter value to add to queue: 20
20 added to queue.

Do you want to continue (Y/N): y
Enter option (1-3): 1

Enter value to add to queue: 30
30 added to queue.

Do you want to continue (Y/N): y
Enter option (1-3): 3

Display Queue:
  10 20 30

Do you want to continue (Y/N): y
Enter option (1-3): 2
10 deleted from queue.

Do you want to continue (Y/N): y
Enter option (1-3): 3

Display Queue:
  20 30

Do you want to continue (Y/N): n

Program terminated successfully...

PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C>
```

Lab 8

Program:

Write a program to take a string of parenthesis as input and to check whether the parenthesis are balanced or not with the use of stacks.

Source Code:

```
#include<stdio.h>
#include<string.h>
#define MAX 100
void parenthesis_check(char s[]) {
    char stack[MAX];
    int stack_index = -1;
    int len = strlen(s);
    char input[MAX];
    int input_index = -1;
    int flag = 0;
    for(int i = 0; i < len; i++) {
        char c = s[i];
        if(c == '(' || c == '[' || c == '{') {
            stack_index++;
            stack[stack_index]  = c;
        }
        else if(c == ')' || c == ']' || c == '}') {
            if(stack_index > -1) {
                if((c == ')' && stack[stack_index] == '(') || (c == ']' &&
stack[stack_index] == '[') || (c == '}' && stack[stack_index] == '{')) {
                    stack_index--;
                    // printf("%d \n", stack_index);
                }
                else {
                    // printf("Unbalanced!\n");
                    flag = 1;
    break;
                }
            }
            else if(stack_index == -1) {
                // printf("Unbalanced!\n");
                flag = 1;
    break;
            }
        }
    }
    if(stack_index == -1) {
        if(flag == 0) {
            printf("\nParenthesis string is balanced...\n\n");
        }
```

```
        else {
            printf("\nStirng is unbalanced...\n\n");
        }
    }
    else if(stack_index >= 0) {
        printf("\nString is unbalanced...\n\n");
    }
}
int main() {
    char string[MAX];
    printf("Parenthesis Checker:\n\n");
    printf("Enter parenthesis string: ");
    fgets(string, MAX, stdin);
    parenthesis_check(string);
    return 0;
}
```

Output:

```
Parenthesis Checker:

Enter parenthesis string: {()[([])[]]}

Parenthesis string is balanced...

PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C> cd "c:\Users\Faizan\Deskt
sis_Checker.c -o Parenthesis_Checker } ; if ($?) { .\Parenthesis_Checker }
Parenthesis Checker:

Enter parenthesis string: {{()[([])[]]}}

String is unbalanced...

PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C>
```

Lab 9

Program:

Write a program to perform infix to prefix conversion on an expression and to perform evaluation of the prefix expression.

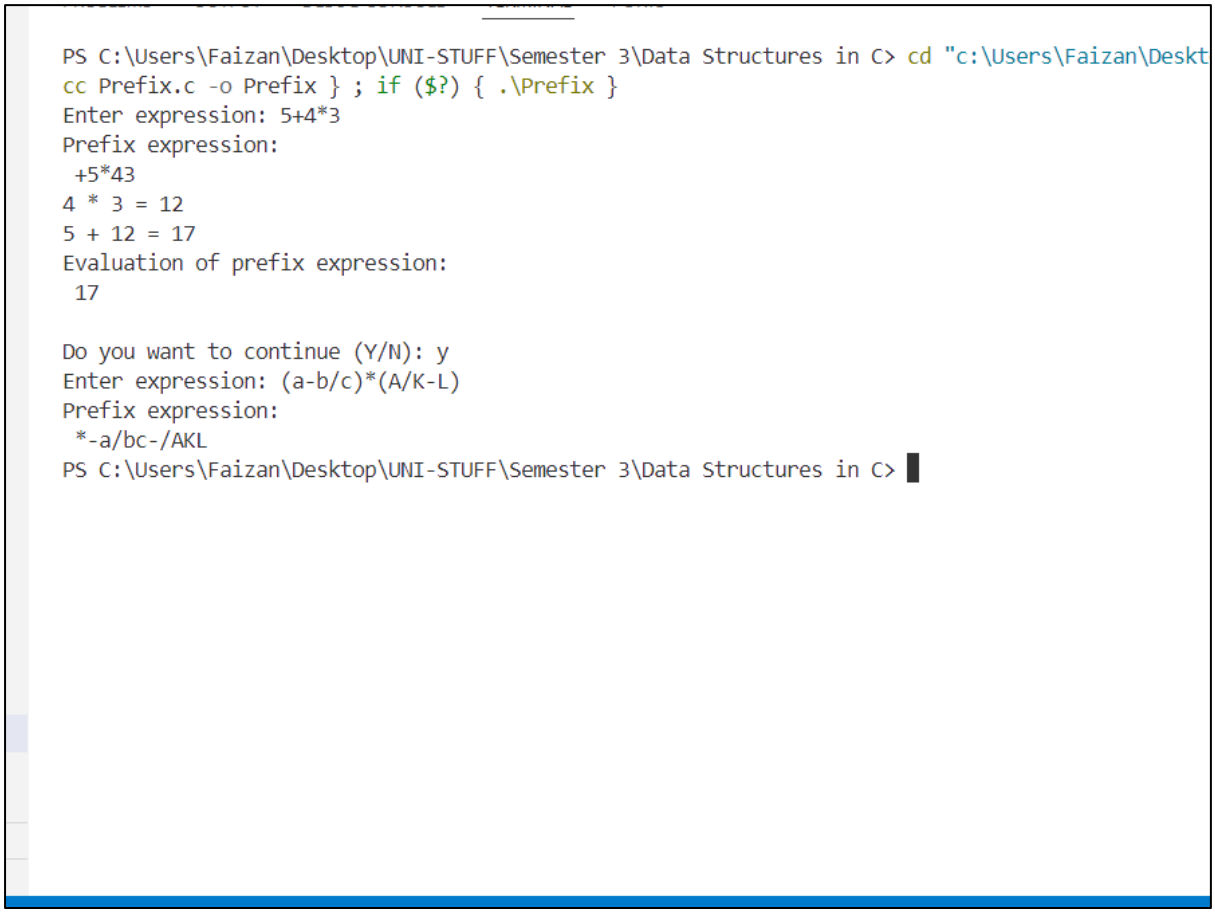
Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define MAX 100
int precedence(char c) {
    if (c == '^')
        return 3;
    else if (c == '/' || c == '*')
        return 2;
    else if (c == '+' || c == '-')
        return 1;
    else
        return -1;
}
void ReverseString(char* str) {
    int len = strlen(str);
    for (int i = 0; i < len / 2; i++) {
        char temp = str[i];
        str[i] = str[len - 1 - i];
        str[len - 1 - i] = temp;
    }
}
void Prefix(char s[]) {
    char exp[MAX];
    int len = strlen(s);
    char stack[MAX];
    int stack_index = -1;
    int output_index = 0;
    ReverseString(s);
    for (int i = 0; i < len; i++) {
        char c = s[i];
        if (isdigit(c) || isalpha(c)) {
            exp[output_index++] = c;
        } else if (c == ')') {
            stack[++stack_index] = c;
        } else if (c == '(') {
            while (stack_index >= 0 && stack[stack_index] != ')') {
                exp[output_index++] = stack[stack_index--];
            }
```

```
            stack_index--;
        } else {
            while (stack_index >= 0 && precedence(c) <
precedence(stack[stack_index])) {
                exp[output_index++] = stack[stack_index--];
            }
            stack[++stack_index] = c;
        }
    }
    while (stack_index >= 0) {
        exp[output_index++] = stack[stack_index--];
    }
    exp[output_index] = '\0';
    ReverseString(exp);
    printf("Prefix expression:\n %s\n", exp);
    int evaluation[MAX];
    int eval_index = -1;
    for (int i = strlen(exp) - 1; i >= 0; i--) {
        char e = exp[i];
        if (isdigit(e)) {
            evaluation[++eval_index] = e - '0';
        } else if (e == '+' || e == '-' || e == '*' || e == '/') {
            int a = evaluation[eval_index--];
            int b = evaluation[eval_index--];
            int result = 0;
            if (e == '+') {
                result = a + b;
                printf("%d + %d = %d\n", a, b, result);
            } else if (e == '-') {
                result = a - b;
                printf("%d - %d = %d\n", a, b, result);
            } else if (e == '*') {
                result = a * b;
                printf("%d * %d = %d\n", a, b, result);
            } else if (e == '/') {
                result = a / b;
                printf("%d / %d = %d\n", a, b, result);
            }
            evaluation[++eval_index] = result;
        }
    }
    printf("Evaluation of prefix expression:\n %d\n", evaluation[eval_index]);
}
int main() {
    char exp[MAX];
    char contn;
    do {
        printf("Enter expression: ");
```

```
        fgets(exp, MAX, stdin);
        exp[strcspn(exp, "\n")] = '\0';
        Prefix(exp);
        int ch;
        while ((ch = getchar()) != '\n' && ch != EOF);
        printf("Do you want to continue (Y/N): ");
        scanf(" %c", &contn);
        while ((ch = getchar()) != '\n' && ch != EOF);
    }
    while(contn == 'y' || contn == 'Y');
    printf("\nProgram terminated successfully...\n\n");
    return 0;
}
```

Output:



Lab 10

Program:

Write a program to perform infix to postfix conversion on an expression and to perform evaluation of the postfix expression.

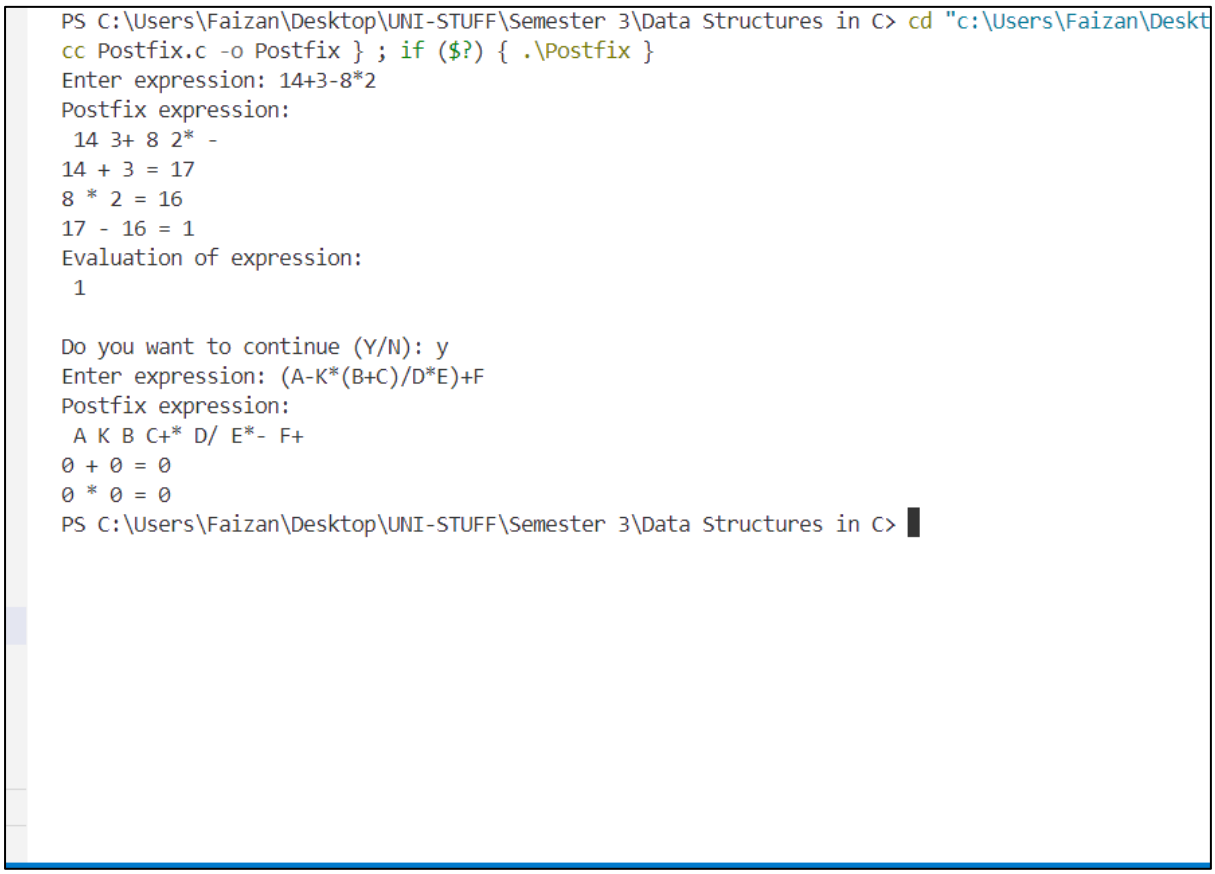
Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define MAX 100
int precedence(char c) {
    if (c == '^')
        return 3;
    else if (c == '/' || c == '*')
        return 2;
    else if (c == '+' || c == '-')
        return 1;
    else
        return -1;
}
void Postfix(char s[]) {
    char output_exp[MAX];
    int output_index = 0;
    int len = strlen(s);
    char stack[MAX];
    int stack_index = -1;
    for (int i = 0; i < len; i++) {
        char c = s[i];
        if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || (c >= '0' && c
<= '9')) {
            output_exp[output_index++] = c;
        }
        else if (c == '(') {
            stack[++stack_index] = c;
        }
        else if (c == ')') {
            while (stack_index >= 0 && stack[stack_index] != '(') {
                output_exp[output_index++] = stack[stack_index--];
            }
            stack_index--;
        }
        else {
            while (stack_index >= 0 && (precedence(s[i]) <
precedence(stack[stack_index]) ||
```

```
                                        precedence(s[i]) ==
precedence(stack[stack_index]))) {
                output_exp[output_index++] = stack[stack_index--];
            }
            output_exp[output_index++] = ' ';
            stack[++stack_index] = c;
        }
    }
    while (stack_index >= 0) {
        output_exp[output_index++] = stack[stack_index--];
        output_exp[output_index++] = ' ';
    }
    output_exp[output_index] = '\0';
    printf("Postfix expression:\n %s\n", output_exp);
    int evaluation[MAX];
    int eval_index = -1;
    int i = 0;
    while (output_exp[i] != '\0') {
        if (isdigit(output_exp[i])) {
            int num = 0;
            while (isdigit(output_exp[i])) {
                num = num * 10 + (output_exp[i] - '0');
                i++;
            }
            evaluation[++eval_index] = num;
        } else if (output_exp[i] == ' ') {
            i++;
        } else if (output_exp[i] == '+' || output_exp[i] == '-' ||
output_exp[i] == '*' || output_exp[i] == '/') {
            int b = evaluation[eval_index--];
            int a = evaluation[eval_index--];
            int result = 0;
            if (output_exp[i] == '+') {
                result = a + b;
                printf("%d + %d = %d\n", a, b, result);
            } else if (output_exp[i] == '-') {
                result = a - b;
                printf("%d - %d = %d\n", a, b, result);
            } else if (output_exp[i] == '*') {
                result = a * b;
                printf("%d * %d = %d\n", a, b, result);
            } else if (output_exp[i] == '/') {
                result = a / b;
                printf("%d / %d = %d\n", a, b, result);
            }
            evaluation[++eval_index] = result;
            i++;
        } else {
```

```
            i++;
        }
    }
    printf("Evaluation of expression:\n %d\n", evaluation[eval_index]);
}
int main() {
    char exp[MAX];
    char contn;
    int ch;
    do {
        printf("Enter expression: ");
        fgets(exp, MAX, stdin);
        exp[strcspn(exp, "\n")] = '\0';
        Postfix(exp);
        while ((ch = getchar()) != '\n' && ch != EOF);
        printf("Do you want to continue (Y/N): ");
        scanf(" %c", &contn);
        while ((ch = getchar()) != '\n' && ch != EOF);
    }
    while (contn == 'y' || contn == 'Y');
    printf("\nProgram terminated successfully...\n\n");
    return 0;
}
```

Output:



Lab 11

Program:

Write a menu driven program to perform the operations of Tower of Hanoi, Factorial, and Fibonacci.

Source code:

```
#include<stdio.h>
int step = 1;
void TOH(int n, char S, char D, char A){
    if(n == 1) {
        printf("Step %d: Move disk from tower %d to tower %d\n", step++, S,
D);
    }
    else{
        TOH(n-1, S, A, D);
        printf("Step %d: Move disk from tower %d to tower %d.\n", step++, S,
D);
        TOH(n-1, A, D, S);
    }
}
int Factorial(int n){
    if(n == 0 || n == 1)
        return 1;
    else
        return n * Factorial(n-1);
}
void Fibonacci(int n){
    int a = 0, b = 1, next;
    if(n == 0) {
        printf("Enter a positive number.\n");
        return;
    }
    printf("Fibonacci for %d terms:\n", n);
    for(int i = 0; i < n; i++) {
        if(i == 0)
            printf("%d ", a);
        else if(i == 1)
            printf("%d ", b);
        else {
            next = a + b;
            a = b;
            b = next;
            printf("%d ", next);
        }
    }
    printf("\n");
```

```
}
int main(){
    int n, optn;
    int Tsource, Tdest, Taux;
    char contn;
    printf("Operations:\n 1- Tower of Hanoi.\n 2- Factorial.\n 3-
Fibonacci.\n");
    do {
        printf("Enter option: ");
        scanf("%d", &optn);
        switch(optn) {
            case 1:
                printf("Enter number of rings: ");
                scanf("%d", &n);
                printf("Enter source tower: ");
                scanf("%d", &Tsource);
                printf("Enter destination tower: ");
                scanf("%d", &Tdest);
                TOH(n, Tsource, Tdest, Taux);
                printf("Total steps: %d\n", step-1);
                break;
            case 2:
                printf("Enter number for factorial: ");
                scanf("%d", &n);
                printf("Factorial of %d is: %d", n, Factorial(n));
                break;
            case 3:
                printf("Enter number of terms in Fibonacci: ");
                scanf("%d", &n);
                Fibonacci(n);
                break;
            default:
                printf("\nIncorrect option.\nChoose from given options(1-
3).\n");
                break;
        }
        printf("\nDo you want to continue(Y/N): ");
        scanf(" %c", &contn);
    }
    while(contn == 'Y' || contn == 'y');
    printf("\nProgram terminated successfully...\n\n");
    return 0;
}
```

Output:

```
 TOH } ; if ($?) { .\TOH }
Operations:
 1- Tower of Hanoi.
 2- Factorial.
 3- Fibonacci.
Enter option: 1
Enter number of rings: 3
Enter source tower: 1
Enter destination tower: 3
Step 1: Move disk from tower 1 to tower 3
Step 2: Move disk from tower 1 to tower 0.
Step 3: Move disk from tower 3 to tower 0
Step 4: Move disk from tower 1 to tower 3.
Step 5: Move disk from tower 0 to tower 1
Step 6: Move disk from tower 0 to tower 3.
Step 7: Move disk from tower 1 to tower 3
Total steps: 7

Do you want to continue(Y/N): y
Enter option: 2
Enter number for factorial: 7
Factorial of 7 is: 5040
Do you want to continue(Y/N): y
Enter option: 3
Enter number of terms in Fibonacci: 10
Fibonacci for 10 terms:
0 1 1 2 3 5 8 13 21 34

Do you want to continue(Y/N): n

Program terminated successfully...

PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C>
```

Lab 12.1

Program:

Write a program to perform Bubble sorting on an array of size n. Take the elements and size of array from user.

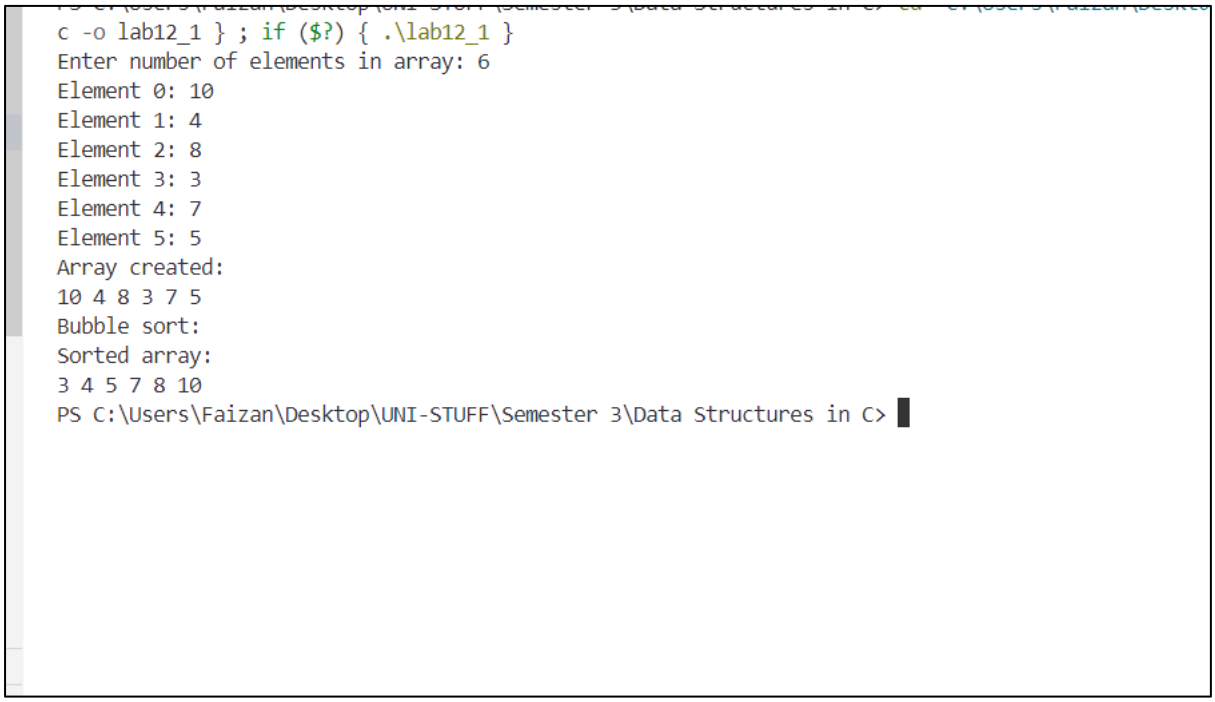
Source code:

```c
#include<stdio.h>
#define MAX 100
void bubble_sort(int n, int a[MAX]){
    for(int i = 0; i < n; i++){
        for(int j = 0; j < (n-1); j++){
            if(a[j] > a[j+1]) {
                int temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }
        }
    }
    printf("Sorted array:\n");
    for(int i = 0; i < n; i++){
        printf("%d ", a[i]);
    }
}
int main(){
    int n, arr[MAX];
    printf("Enter number of elements in array: ");
    scanf("%d", &n);
    for(int i = 0; i < n; i++){
        printf("Element %d: ", i);
        scanf("%d", &arr[i]);
    }
    printf("Array created:\n");
    for(int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\nBubble sort:\n");

    bubble_sort(n, arr);


    return 0;
}
```

Output:


c -o lab12_1 } ; if ($?) { .\lab12_1 }
Enter number of elements in array: 6
Element 0: 10
Element 1: 4
Element 2: 8
Element 3: 3
Element 4: 7
Element 5: 5
Array created:
10 4 8 3 7 5
Bubble sort:
Sorted array:
3 4 5 7 8 10
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C>

Lab 12.2

Program:

Write a program to perform Selection sorting on an array of size n. Take the elements and size of array from user.

Source code:

```c
#include <stdio.h>
#define MAX 100
void selectionSort(int arr[MAX], int n) {
    for (int i = 0; i < n - 1; i++) {
        int min = i;
        for (int j = i + 1; j < n; j++) {
            if (arr[j] < arr[min]) {
                min = j;
            }
        }
        int temp = arr[min];
        arr[min] = arr[i];
        arr[i] = temp;
    }
}
int main() {
    int arr[MAX];
    int n;
    printf("Enter n: ");
    scanf("%d", &n);
    printf("Create array:\n");
    for(int i = 0; i < n; i++){
        printf("Element %d: ", i);
        scanf("%d", &arr[i]);
    }
    printf("Unsorted array: \n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    selectionSort(arr, n);
    printf("Sorted array: \n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

Output:

```
cc lab12_2.c -o lab12_2 } ; if ($?) { .\lab12_2 }
Enter n: 6
Create array:
Element 0: 15
Element 1: 7
Element 2: 6
Element 3: 9
Element 4: 4
Element 5: 12
Unsorted array:
15 7 6 9 4 12
Sorted array:
4 6 7 9 12 15
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C>
```

Lab 12.3

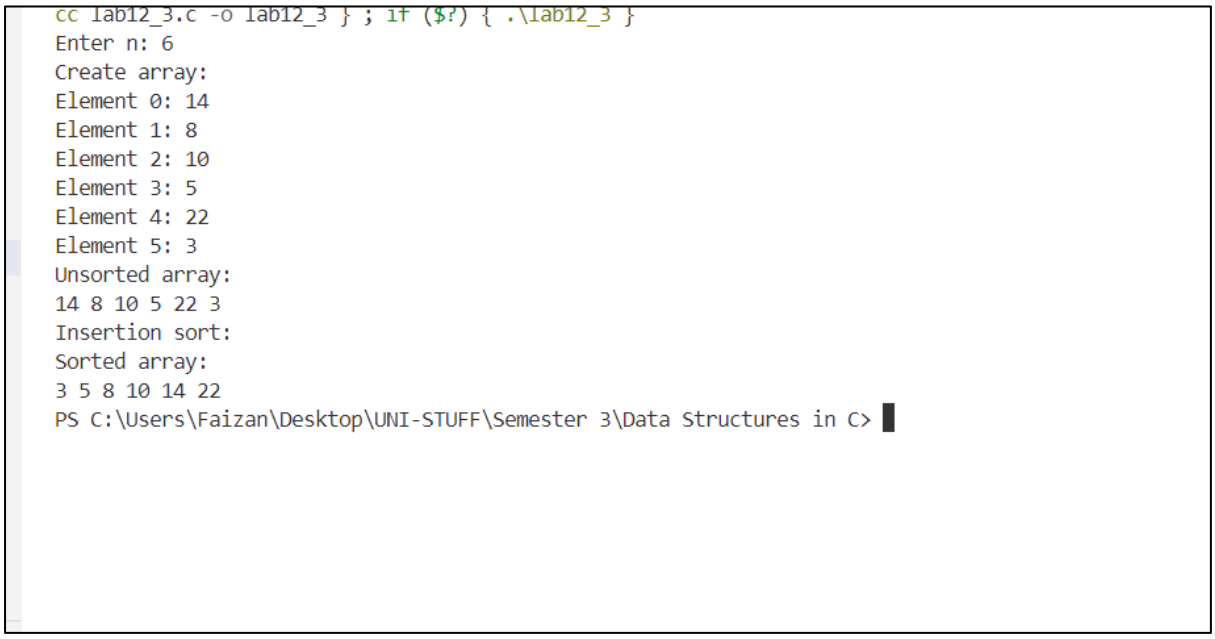
Program:

Write a program to perform Insertion sorting on an array of size n. Take the elements and size of array from user.

Source code:

```c
#include<stdio.h>
#define MAX 100
void insertionSort(int n, int arr[MAX]){
    for(int i = 0; i < n; i++){
        int key = arr[i];
        int j = i-1;
        while(j >= 0 && arr[j] > key){
            arr[j+1] = arr[j];
            j = j - 1;
        }
        arr[j+1] = key;
    }
}
int main(){
    int n, arr[MAX];
    printf("Enter n: ");
    scanf("%d", &n);
    printf("Create array:\n");
    for(int i = 0; i < n; i++){
        printf("Element %d: ", i);
        scanf("%d", &arr[i]);
    }
    printf("Unsorted array: \n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("Insertion sort:\n");
    insertionSort(n, arr);
    printf("Sorted array: \n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

Output:


cc lab12_3.c -o lab12_3 } ; if ($?) { .\lab12_3 }
Enter n: 6
Create array:
Element 0: 14
Element 1: 8
Element 2: 10
Element 3: 5
Element 4: 22
Element 5: 3
Unsorted array:
14 8 10 5 22 3
Insertion sort:
Sorted array:
3 5 8 10 14 22
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C>

Lab 13.1

Program:

Write a program to perform Insertion and Deletion from the beginning, middle, and the end of a Singly Linked List.

Source code:

```
#include<stdio.h>
#include<stdlib.h>
typedef struct Node {
    int data;
    struct Node* next;
} Node;
Node* createNode(int data) {
    Node* NN = (Node*)malloc(sizeof(Node));
    if(NN == NULL) {
        printf("List overflow...\n");
        exit(1);
    }
    NN->data = data;
    NN->next = NULL;
    return NN;
}

void insert_beginning(Node** start, int data) {
    Node* NN = createNode(data);
    NN->next = *start;
    *start = NN;
}

void insert_middle(Node** start, int data, int position) {
    if(position < 0){
        printf("Invalid position...\n");
        return;
    }
    if(position == 0){
        insert_beginning(start, data);
        return;
    }
    Node* NN = createNode(data);
    Node* temp = *start;
    for(int i = 0; temp != NULL && i < (position-1); i++)
    {
        temp = temp->next;
    }
    if(temp == NULL) {
        printf("Incorrect position...\n");
```

```
        free(NN);
        return;
    }
    NN->next = temp->next;
    temp->next = NN;
}

void insert_end(Node** start, int data) {
    Node* NN = createNode(data);
    if(*start == NULL) {
        *start = NN;
        return;
    }
    Node* end = *start;
    while(end->next != NULL) {
        end = end->next;
    }
    end->next = NN;
}

void delete_beginning(Node** start) {
    if(*start == NULL){
        printf("List underflow.\n");
        return;
    }
    Node* temp = *start;
    *start = temp->next;
    free(temp);
}

void delete_middle(Node** start, int position) {
    if(*start == NULL){
        printf("List underflow.\n");
        return;
    }
    if(position == 0){
        delete_beginning(start);
        return;
    }
    Node* temp = *start;
    for (int i = 0; temp != NULL && i < position - 1; i++) {
        temp = temp->next;
    }
    if (temp == NULL || temp->next == NULL) {
        printf("Incorrect position.\n");
        return;
    }
    Node* NN = temp->next;
```

```
    temp->next = NN->next;
    free(NN);
}

void delete_end(Node** start) {
    if(*start == NULL) {
        printf("List underflow.\n");
        return;
    }
    if((*start)->next == NULL) {
        free(*start);
        *start = NULL;
        return;
    }
    Node* temp = *start;
    while((temp->next)->next != NULL) {
        temp = temp->next;
    }
    free(temp->next);
    temp->next = NULL;
}

void display(Node* node) {
    while (node != NULL) {
        printf("%d -> ", node->data);
        node = node->next;
    }
    printf("NULL\n");
}

int main()
{
    Node* start = NULL;
    int val, position, optn;
    char contn;

    printf("Linked List:\n Insert:\n  1- Beginning.\n  2- Middle.\n  3- End.\n
Delete:\n  4- Beginning.\n  5- Middle.\n  6- End.\n  7- Display.\n\n");

    do {
        printf("Enter option: ");
        scanf("%d", &optn);

        switch(optn){
            case 1:
                printf("Enter value: ");
                scanf("%d", &val);
                insert_beginning(&start, val);
```

```
                break;
            case 2:
                printf("Enter value: ");
                scanf("%d", &val);
                printf("Enter position: ");
                scanf("%d", &position);
                insert_middle(&start, val, position);
                break;
            case 3:
                printf("Enter value: ");
                scanf("%d", &val);
                insert_end(&start, val);
                break;
            case 4:
                delete_beginning(&start);
                break;
            case 5:
                printf("Enter position: ");
                scanf("%d", &position);
                delete_middle(&start, position);
                break;
            case 6:
                delete_end(&start);
                break;
            case 7:
                printf("Linked List display:\n");
                display(start);
                break;
            default:
                printf("Incorrect option...\n");
                break;
        }
        printf("Do you want to continue(Y/N): ");
        scanf(" %c", &contn);
    }
    while(contn == 'Y' || contn == 'y');

    printf("\nProgram terminated successfully...\n");

    return 0;
}
```

Output:

```
{ .\SingleLinkedList }
Linked List:
 Insert:
  1- Beginning.
  2- Middle.
  3- End.
 Delete:
  4- Beginning.
  5- Middle.
  6- End.
  7- Display.

Enter option: 1
Enter value: 10
Do you want to continue(Y/N): y
Enter option: 3
Enter value: 20
Do you want to continue(Y/N): y
Enter option: 2
Enter value: 15
Enter position: 1
Do you want to continue(Y/N): y
Enter option: 7
Linked List display:
10 -> 15 -> 20 -> NULL
Do you want to continue(Y/N): y
Enter option: 5
Enter position: 1
Do you want to continue(Y/N): y
Enter option: 7
Linked List display:
10 -> 20 -> NULL
Do you want to continue(Y/N): y
Enter option: 6
Do you want to continue(Y/N): y
Enter option: 7
Linked List display:
10 -> NULL
Do you want to continue(Y/N): y
Enter option: 4
Do you want to continue(Y/N): y
Enter option: 7
Linked List display:
NULL
Do you want to continue(Y/N): n

Program terminated successfully...
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C>
```

Lab 13.2

Program:

Write a program to perform Insertion and Deletion from the beginning, middle, and the end of a Doubly Linked List.

Source code:

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
} Node;
Node* createNode(int data) {
    Node* NN = (Node*)malloc(sizeof(Node));
    if (NN == NULL) {
        printf("List overflow...\n");
        exit(1);
    }
    NN->data = data;
    NN->next = NULL;
    NN->prev = NULL;
    return NN;
}
void insert_beginning(Node** start, int data) {
    Node* NN = createNode(data);
    NN->next = *start;
    if (*start != NULL) {
        (*start)->prev = NN;
    }
    *start = NN;
}
void insert_middle(Node** start, int data, int position) {
    if (position < 0) {
        printf("Invalid position...\n");
        return;
    }
    if (position == 0) {
        insert_beginning(start, data);
        return;
    }
    Node* NN = createNode(data);
    Node* temp = *start;
    for (int i = 0; temp != NULL && i < (position - 1); i++) {
        temp = temp->next;
    }
```

```
    if (temp == NULL) {
        printf("Incorrect position...\n");
        free(NN);
        return;
    }
    NN->next = temp->next;
    NN->prev = temp;
    if (temp->next != NULL) {
        temp->next->prev = NN;
    }
    temp->next = NN;
}

void insert_end(Node** start, int data) {
    Node* NN = createNode(data);
    if (*start == NULL) {
        *start = NN;
        return;
    }
    Node* end = *start;
    while (end->next != NULL) {
        end = end->next;
    }
    end->next = NN;
    NN->prev = end;
}

void delete_beginning(Node** start) {
    if (*start == NULL) {
        printf("List underflow.\n");
        return;
    }
    Node* temp = *start;
    *start = temp->next;
    if (*start != NULL) {
        (*start)->prev = NULL;
    }
    free(temp);
}

void delete_middle(Node** start, int position) {
    if (*start == NULL) {
        printf("List underflow.\n");
        return;
    }
    if (position == 0) {
        delete_beginning(start);
        return;
```

```
    }
    Node* temp = *start;
    for (int i = 0; temp != NULL && i < position; i++) {
        temp = temp->next;
    }
    if (temp == NULL) {
        printf("Incorrect position.\n");
        return;
    }
    if (temp->prev != NULL) {
        temp->prev->next = temp->next;
    }
    if (temp->next != NULL) {
        temp->next->prev = temp->prev;
    }
    free(temp);
}
void delete_end(Node** start) {
    if (*start == NULL) {
        printf("List underflow.\n");
        return;
    }
    if ((*start)->next == NULL) {
        free(*start);
        *start = NULL;
        return;
    }
    Node* temp = *start;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->prev->next = NULL;
    free(temp);
}
void display(Node* node) {
    while (node != NULL) {
        printf("%d <-> ", node->data);
        node = node->next;
    }
    printf("NULL\n");
}
int main() {
    Node* start = NULL;
    int val, position, optn;
    char contn;

    printf("Doubly Linked List:\n Insert:\n  1- Beginning.\n  2- Middle.\n  3-
End.\n Delete:\n  4- Beginning.\n  5- Middle.\n  6- End.\n  7- Display.\n\n");
```

```
    do {
        printf("Enter option: ");
        scanf("%d", &optn);
        switch (optn) {
            case 1:
                printf("Enter value: ");
                scanf("%d", &val);
                insert_beginning(&start, val);
                break;
            case 2:
                printf("Enter value: ");
                scanf("%d", &val);
                printf("Enter position: ");
                scanf("%d", &position);
                insert_middle(&start, val, position);
                break;
            case 3:
                printf("Enter value: ");
                scanf("%d", &val);
                insert_end(&start, val);
                break;
            case 4:
                delete_beginning(&start);
                break;
            case 5:
                printf("Enter position: ");
                scanf("%d", &position);
                delete_middle(&start, position);
                break;
            case 6:
                delete_end(&start);
                break;
            case 7:
                printf("Doubly Linked List display:\n");
                display(start);
                break;
            default:
                printf("Incorrect option...\n");
                break;
        }
        printf("Do you want to continue(Y/N): ");
        scanf(" %c", &contn);
    } while (contn == 'Y' || contn == 'y');
    printf("\nProgram terminated successfully...\n");
    return 0;
}
```

Output:

```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C> cd "c:\Users\Faizan\Deskto
st } ; if ($?) { .\DoubleLinkedList }
Doubly Linked List:
 Insert:
  1- Beginning.
  2- Middle.
  3- End.
 Delete:
  4- Beginning.
  5- Middle.
  6- End.
  7- Display.

Enter option: 1
Enter value: 10
Do you want to continue(Y/N): y
Enter option: 3
Enter value: 20
Do you want to continue(Y/N): y
Enter option: 2
Enter value: 15
Enter position: 1
Do you want to continue(Y/N): y
Enter option: 7
Doubly Linked List display:
10 <-> 15 <-> 20 <-> NULL
Do you want to continue(Y/N): y
Enter option: 5
Enter position: 1
Do you want to continue(Y/N): y
Enter option: 7
Doubly Linked List display:
10 <-> 20 <-> NULL
Do you want to continue(Y/N): y
Enter option: 6
Do you want to continue(Y/N): y
Enter option: 7
Doubly Linked List display:
10 <-> NULL
Do you want to continue(Y/N): y
Enter option: 4
Do you want to continue(Y/N): y
Enter option: 7
Doubly Linked List display:
NULL
Do you want to continue(Y/N): n

Program terminated successfully...
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C>
```

Lab 14.1

Program:

Write a program to perform Insertion and Deletion from the beginning, middle, and the end of a circular Singly Linked List.

Source code:

```
#include<stdio.h>
#include<stdlib.h>
typedef struct Node {
    int data;
    struct Node* next;
} Node;
Node* createNode(int data) {
    Node* NN = (Node*)malloc(sizeof(Node));
    if(NN == NULL) {
        printf("List overflow...\n");
        exit(1);
    }
    NN->data = data;
    NN->next = NULL;
    return NN;
}
void insert_beginning(Node** start, int data) {
    Node* NN = createNode(data);
    if(*start == NULL) {
        *start = NN;
        NN->next = *start;
    } else {
        Node* temp = *start;
        while(temp->next != *start) {
            temp = temp->next;
        }
        temp->next = NN;
        NN->next = *start;
        *start = NN;
    }
}
void insert_middle(Node** start, int data, int position) {
    if(position < 0) {
        printf("Invalid position...\n");
        return;
    }
    if(position == 0) {
        insert_beginning(start, data);
        return;
    }
```

```
    Node* NN = createNode(data);
    Node* temp = *start;
    for(int i = 0; i < position - 1 && temp->next != *start; i++) {
        temp = temp->next;
    }
    if(temp->next == *start && position > 1) {
        printf("Incorrect position...\n");
        free(NN);
        return;
    }
    NN->next = temp->next;
    temp->next = NN;
}
void insert_end(Node** start, int data) {
    Node* NN = createNode(data);
    if(*start == NULL) {
        *start = NN;
        NN->next = *start;
    } else {
        Node* temp = *start;
        while(temp->next != *start) {
            temp = temp->next;
        }
        temp->next = NN;
        NN->next = *start;
    }
}
void delete_beginning(Node** start) {
    if(*start == NULL) {
        printf("List underflow.\n");
        return;
    }
    Node* temp = *start;
    if(temp->next == *start) {
        free(temp);
        *start = NULL;
    } else {
        Node* last = *start;
        while(last->next != *start) {
            last = last->next;
        }
        *start = temp->next;
        last->next = *start;
        free(temp);
    }
}
void delete_middle(Node** start, int position) {
    if(*start == NULL) {
```

```
        printf("List underflow.\n");
        return;
    }
    if(position == 0) {
        delete_beginning(start);
        return;
    }
    Node* temp = *start;
    for(int i = 0; i < position - 1 && temp->next != *start; i++) {
        temp = temp->next;
    }
    if(temp->next == *start) {
        printf("Incorrect position.\n");
        return;
    }
    Node* NN = temp->next;
    temp->next = NN->next;
    free(NN);
}
void delete_end(Node** start) {
    if(*start == NULL) {
        printf("List underflow.\n");
        return;
    }
    Node* temp = *start;
    if(temp->next == *start) {
        free(temp);
        *start = NULL;
        return;
    }
    Node* prev = NULL;
    while(temp->next != *start) {
        prev = temp;
        temp = temp->next;
    }
    free(temp);
    prev->next = *start;
}
void display(Node* node) {
    if(node == NULL) {
        printf("List is empty.\n");
        return;
    }
    Node* temp = node;
    do {
        printf("%d -> ", temp->data);
        temp = temp->next;
    } while(temp != node);
```

```
    printf("(back to start)\n");
}
int main() {
    Node* start = NULL;
    int val, position, optn;
    char contn;
    printf("Linked List:\n Insert:\n  1- Beginning.\n  2- Middle.\n  3- End.\n
Delete:\n  4- Beginning.\n  5- Middle.\n  6- End.\n  7- Display.\n\n");
    do {
        printf("Enter option: ");
        scanf("%d", &optn);
        switch(optn) {
            case 1:
                printf("Enter value: ");
                scanf("%d", &val);
                insert_beginning(&start, val);
                break;
            case 2:
                printf("Enter value: ");
                scanf("%d", &val);
                printf("Enter position: ");
                scanf("%d", &position);
                insert_middle(&start, val, position);
                break;
            case 3:
                printf("Enter value: ");
                scanf("%d", &val);
                insert_end(&start, val);
                break;
            case 4:
                delete_beginning(&start);
                break;
            case 5:
                printf("Enter position: ");
                scanf("%d", &position);
                delete_middle(&start, position);
                break;
            case 6:
                delete_end(&start);
                break;
            case 7:
                printf("Linked List display:\n");
                display(start);
                break;
            default:
                printf("Incorrect option...\n");
                break;
        }
```

```
        printf("Do you want to continue(Y/N): ");
        scanf(" %c", &contn);
    } while(contn == 'Y' || contn == 'y');
    printf("\nProgram terminated successfully...\n");
    return 0;
}
```

Output:

```
Linked List:
 Insert:
  1- Beginning.
  2- Middle.
  3- End.
 Delete:
  4- Beginning.
  5- Middle.
  6- End.
  7- Display.

Enter option: 1
Enter value: 10
Do you want to continue(Y/N): y
Enter option: 3
Enter value: 20
Do you want to continue(Y/N): y
Enter option: 3
Enter value: 40
Do you want to continue(Y/N): y
Enter option: 2
Enter value: 30
Enter position: 2
Do you want to continue(Y/N): y
Enter option: 7
Linked List display:
10 -> 20 -> 30 -> 40 -> (back to start)
Do you want to continue(Y/N): y
Enter option: 5
Enter position: 1
Do you want to continue(Y/N): y
Enter option: 7
Linked List display:
10 -> 30 -> 40 -> (back to start)
Do you want to continue(Y/N): y
Enter option: 6
Do you want to continue(Y/N): y
Enter option: 4
Do you want to continue(Y/N): y
Enter option: 7
Linked List display:
30 -> (back to start)
Do you want to continue(Y/N): y
Enter option: 4
Do you want to continue(Y/N): y
Enter option: 7
Linked List display:
List is empty.
Do you want to continue(Y/N): n

Program terminated successfully...
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C>
```

Lab 14.2

Program:

Write a program to perform Insertion and Deletion from the beginning, middle, and the end of a circular Doubly Linked List.

Source code:

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
} Node;
Node* createNode(int data) {
    Node* NN = (Node*)malloc(sizeof(Node));
    if (NN == NULL) {
        printf("List overflow...\n");
        exit(1);
    }
    NN->data = data;
    NN->next = NN;
    NN->prev = NN;
    return NN;
}
void insert_beginning(Node** start, int data) {
    Node* NN = createNode(data);
    if (*start == NULL) {
        *start = NN;
    } else {
        Node* last = (*start)->prev;
        NN->next = *start;
        NN->prev = last;
        last->next = NN;
        (*start)->prev = NN;
    }
    *start = NN;
}
void insert_middle(Node** start, int data, int position) {
    if (position < 0) {
        printf("Invalid position...\n");
        return;
    }
    if (position == 0) {
        insert_beginning(start, data);
        return;
    }
```

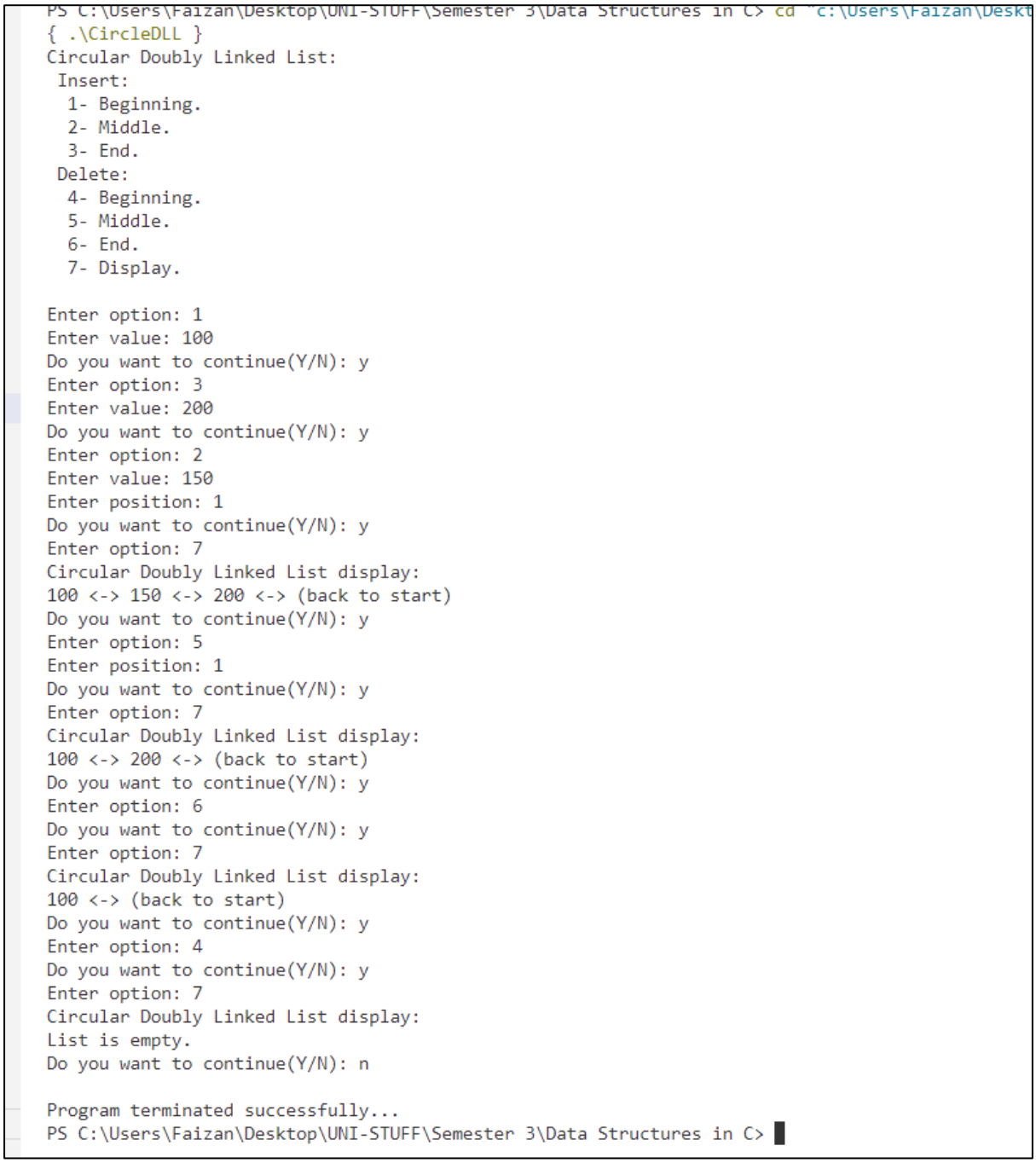
```
    Node* NN = createNode(data);
    Node* temp = *start;
    for (int i = 0; i < position - 1 && temp->next != *start; i++) {
        temp = temp->next;
    }
    if (temp->next == *start && position > 1) {
        printf("Incorrect position...\n");
        free(NN);
        return;
    }
    NN->next = temp->next;
    NN->prev = temp;
    temp->next->prev = NN;
    temp->next = NN;
}
void insert_end(Node** start, int data) {
    if (*start == NULL) {
        insert_beginning(start, data);
    } else {
        Node* NN = createNode(data);
        Node* last = (*start)->prev;
        last->next = NN;
        NN->prev = last;
        NN->next = *start;
        (*start)->prev = NN;
    }
}
void delete_beginning(Node** start) {
    if (*start == NULL) {
        printf("List underflow.\n");
        return;
    }
    Node* temp = *start;
    if (temp->next == *start) {
        free(temp);
        *start = NULL;
    } else {
        Node* last = temp->prev;
        *start = temp->next;
        last->next = *start;
        (*start)->prev = last;
        free(temp);
    }
}
void delete_middle(Node** start, int position) {
    if (*start == NULL) {
        printf("List underflow.\n");
        return;
```

```
    }
    if (position == 0) {
        delete_beginning(start);
        return;
    }
    Node* temp = *start;
    for (int i = 0; i < position && temp->next != *start; i++) {
        temp = temp->next;
    }
    if (temp->next == *start && position > 0) {
        printf("Incorrect position.\n");
        return;
    }
    temp->prev->next = temp->next;
    temp->next->prev = temp->prev;
    free(temp);
}
void delete_end(Node** start) {
    if (*start == NULL) {
        printf("List underflow.\n");
        return;
    }
    Node* last = (*start)->prev;
    if (last == *start) {
        free(last);
        *start = NULL;
    } else {
        Node* secondLast = last->prev;
        secondLast->next = *start;
        (*start)->prev = secondLast;
        free(last);
    }
}
void display(Node* node) {
    if (node == NULL) {
        printf("List is empty.\n");
        return;
    }
    Node* temp = node;
    do {
        printf("%d <-> ", temp->data);
        temp = temp->next;
    } while (temp != node);
    printf("(back to start)\n");
}
int main() {
    Node* start = NULL;
    int val, position, optn;
```

```
    char contn;
    printf("Circular Doubly Linked List:\n Insert:\n  1- Beginning.\n  2-
Middle.\n  3- End.\n Delete:\n  4- Beginning.\n  5- Middle.\n  6- End.\n  7-
Display.\n\n");
    do {
        printf("Enter option: ");
        scanf("%d", &optn);
        switch (optn) {
            case 1:
                printf("Enter value: ");
                scanf("%d", &val);
                insert_beginning(&start, val);
                break;
            case 2:
                printf("Enter value: ");
                scanf("%d", &val);
                printf("Enter position: ");
                scanf("%d", &position);
                insert_middle(&start, val, position);
                break;
            case 3:
                printf("Enter value: ");
                scanf("%d", &val);
                insert_end(&start, val);
                break;
            case 4:
                delete_beginning(&start);
                break;
            case 5:
                printf("Enter position: ");
                scanf("%d", &position);
                delete_middle(&start, position);
                break;
            case 6:
                delete_end(&start);
                break;
            case 7:
                printf("Circular Doubly Linked List display:\n");
                display(start);
                break;
            default:
                printf("Incorrect option...\n");
                break;
        }
        printf("Do you want to continue(Y/N): ");
        scanf(" %c", &contn);
    } while (contn == 'Y' || contn == 'y');
    printf("\nProgram terminated successfully...\n");
```

```
    return 0;
}
```

Output:



```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C> cd "c:\Users\Faizan\Deskt
{ .\CircleDLL }
Circular Doubly Linked List:
 Insert:
  1- Beginning.
  2- Middle.
  3- End.
 Delete:
  4- Beginning.
  5- Middle.
  6- End.
  7- Display.

Enter option: 1
Enter value: 100
Do you want to continue(Y/N): y
Enter option: 3
Enter value: 200
Do you want to continue(Y/N): y
Enter option: 2
Enter value: 150
Enter position: 1
Do you want to continue(Y/N): y
Enter option: 7
Circular Doubly Linked List display:
100 <-> 150 <-> 200 <-> (back to start)
Do you want to continue(Y/N): y
Enter option: 5
Enter position: 1
Do you want to continue(Y/N): y
Enter option: 7
Circular Doubly Linked List display:
100 <-> 200 <-> (back to start)
Do you want to continue(Y/N): y
Enter option: 6
Do you want to continue(Y/N): y
Enter option: 7
Circular Doubly Linked List display:
100 <-> (back to start)
Do you want to continue(Y/N): y
Enter option: 4
Do you want to continue(Y/N): y
Enter option: 7
Circular Doubly Linked List display:
List is empty.
Do you want to continue(Y/N): n

Program terminated successfully...
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C>
```

Lab 15.1

Program:

Write a program to perform Merge sorting on a user defined array of size n.

Source code:

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 100
void merge(int a[], int left, int mid, int right) {
    int i, j, k;
    int n1 = mid - left + 1;
    int n2 = right - mid;
    int arrLeft[n1], arrRight[n2];
    for(i = 0; i < n1; i++) {
        arrLeft[i] = a[left + i];
    }
    for(j = 0; j < n2; j++) {
        arrRight[j] = a[mid + 1 + j];
    }
    i = 0;
    j = 0;
    k = left;
    while(i < n1 && j < n2)
    {
        if(arrLeft[i] <= arrRight[j])
        {
            a[k] = arrLeft[i];
            i++;
        }
        else {
            a[k] = arrRight[j];
            j++;
        }
        k++;
    }

    while(i < n1) {
        a[k] = arrLeft[i];
        i++;
        k++;
    }
    while(j < n2) {
        a[k] = arrRight[j];
        j++;
        k++;
    }
```

```
}

void mergeSort(int a[], int left, int right)
{
    if(left < right) {
        int mid = left + (right - left) / 2;

        mergeSort(a, left, mid);
        mergeSort(a, mid+1, right);
        merge(a, left, mid, right);
    }
}

int main()
{
    int arr[MAX];
    int n;

    printf("Enter numbr of elements of array: ");
    scanf("%d", &n);
    printf("Create array:\n");
    for(int i = 0; i < n; i++)
    {
        printf("Element %d: ", i);
        scanf("%d", &arr[i]);
    }
    printf("Array:\n");
    for(int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
    mergeSort(arr, 0, n-1);
    printf("\nSorted array:\n");
    for(int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

Output:

```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C> cd "c:\User
cc merge.c -o merge } ; if ($?) { .\merge }
Enter numbr of elements of array: 8
Create array:
Element 0: 6
Element 1: 4
Element 2: 8
Element 3: 5
Element 4: 9
Element 5: 7
Element 6: 11
Element 7: 15
Array:
6 4 8 5 9 7 11 15

Sorted array:
4 5 6 7 8 9 11 15
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C>
```

Lab 15.2

Program:

Write a program to perform Quick sorting on a user defined array of size n.

Source code:

```c
#include <stdio.h>

#define MAX 100

void swap(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int partition(int a[], int low, int high) {
    int pivot = a[high];
    int i = (low - 1);

    for (int j = low; j < high; j++) {
        if (a[j] < pivot) {
            i++;
            swap(&a[i], &a[j]);
        }
    }
    swap(&a[i + 1], &a[high]);
    return (i + 1);
}

void quickSort(int a[], int low, int high) {
    if (low < high) {
        int P = partition(a, low, high);
        quickSort(a, low, P-1);
        quickSort(a, P+1, high);
    }
}

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

int main() {
    int arr[MAX];
    int n;
```

```c
    printf("Enter numbr of elements of array: ");
    scanf("%d", &n);
    printf("Create array:\n");
    for(int i = 0; i < n; i++)
    {
        printf("Element %d: ", i);
        scanf("%d", &arr[i]);
    }
    printf("Array:\n");
    for(int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
    quickSort(arr, 0, n - 1);
    printf("\nSorted array:\n");
    printPrray(arr, n);
    return 0;
}
```

Output:

```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C> cd "c:\Users\Faizan\D
cc quick.c -o quick } ; if ($?) { .\quick }
Enter numbr of elements of array: 8
Create array:
Element 0: 19
Element 1: 12
Element 2: 20
Element 3: 7
Element 4: 18
Element 5: 32
Element 6: 5
Element 7: 1
Array:
19 12 20 7 18 32 5 1

Sorted array:
1 5 7 12 18 19 20 32
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C>
```

Lab 15.3

Program:

Write a program to perform Heap sorting on a user defined array of size n.

Source code:

```c
#include <stdio.h>

#define MAX 100

void swap(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

void heapify(int a[], int n, int i) {
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < n && a[left] > a[largest])
        largest = left;

    if (right < n && a[right] > a[largest])
        largest = right;

    if (largest != i) {
        swap(&a[i], &a[largest]);
        heapify(a, n, largest);
    }
}

void heapSort(int a[], int n) {
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(a, n, i);

    for (int i = n - 1; i >= 0; i--) {
        swap(&a[0], &a[i]);
        heapify(a, i, 0);
    }
}

void printArray(int a[], int size) {
    for (int i = 0; i < size; i++)
        printf("%d ", a[i]);
    printf("\n");
```

```
}

int main() {
    int arr[MAX];
    int n;
    printf("Enter numbr of elements of array: ");
    scanf("%d", &n);
    printf("Create array:\n");
    for(int i = 0; i < n; i++)
    {
        printf("Element %d: ", i);
        scanf("%d", &arr[i]);
    }
    printf("Array:\n");
    for(int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
    heapSort(arr, n);
    printf("\nSorted array:\n");
    printArray(arr, n);
    return 0;
}
```

Output:

```
cc heap.c -o heap } ; if ($?) { .\heap }
Enter numbr of elements of array: 15
Create array:
Element 0: 63
Element 1: 19
Element 2: 7
Element 3: 90
Element 4: 81
Element 5: 36
Element 6: 54
Element 7: 48
Element 8: 99
Element 9: 23
Element 10: 20
Element 11: 8
Element 12: 43
Element 13: 60
Element 14: 39
Array:
63 19 7 90 81 36 54 48 99 23 20 8 43 60 39

Sorted array:
7 8 19 20 23 36 39 43 48 54 60 63 81 90 99
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C>
```