VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES TECHNICAL CAMPUS



VIVEKANANDA SCHOOL OF INFORMATION TECHNOLOGY

BACHELOR OF COMPUTER APPLICATIONS

**DATA STRUCTURES AND ALGORITHM USING ‘C’**

**Subject Code - 174**

GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY



***SUBMITTED TO: SUBMITTED BY:***

Dr. Neetu Goel Abhay

Assistant Professor 08117702023

## VSIT-VIPS BCA 2-B

INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Questions** | **Date** | **Signature** |
| 1 | WAP to implement following operations on one dimensional array:   1. Insertion 2. Deletion 3. Traversal 4. Reverse 5. Merge |  |  |
| 2 | WAP to Sort an Array using menu driven:   1. Bubble Sort 2. Merge Sort 3. Insertion sort 4. Selection sort |  |  |
| 3 | WAP to implement a Singly Linked Lists. |  |  |
| 4 | WAP to implement a Circular Linked Lists. |  |  |
| 5 | WAP to implement a Doubly Linked Lists. |  |  |
| 6 | WAP to implement a Static Stack and Dynamic Stack. |  |  |
| 7 | WAP to implement a Static and Dynamic Circular Queue. |  |  |
| 8 | WAP to implement a   1. Static 2. Dynamic De-Queue. |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | Implement recursive algorithms for the following operations on Binary Search Tree:   1. Insertion 2. Searching |  |  |
| 10 | WAP to implement an expression tree. (For Example: (a + b / (c \* d) – e)) |  |  |
| 11 | WAP to search & display the location of an element specified by the user, in an array using:   1. Linear Search 2. Binary Search |  |  |
| 12 | WAP to accept a matrix from user, find out matrix is sparse or not and convert into triplex matrix. |  |  |
| 13 | WAP to implement Polynomial Addition operation using Linked Lists. |  |  |
| 14 | Write a C program to create two linked list from a given list in the following way: INPUT List: 1 2 3 4 5 6 7 8 9 10  OUTPUT:  First List: 1 3 5 7 9  Second List: 2 4 6 8 10 |  |  |
| 15 | WAP to implement Student Database using Linked List with the following structure:   * Name * Roll no * Marks of 5 Subjects * Average * Result, if the average < 50 then print ‘Fail’, otherwise ‘Pass’. |  |  |
| 16 | Write a program to convert Infix to equivalent   1. Prefix expression 2. Postfix expression |  |  |
| **17**  18 | Write a program to evaluate   1. Prefix expression 2. Post expression   Let us assume a Patient's coupon generator for the Doctors’ clinic. The patients are given the coupons on first-come-first- serve basis. After the visit of a patient, patient-ID is kept stack-wise. At the end of the day, the count is generated from the stack. Construct a menu-based program for patients’ coupons generator using an appropriate data structure. |  |  |

**PRACTICAL:6**

WAP to implement static and dynamic stack.

**Code:**

#include<stdio.h>

int stack**[**10**];** int top **=** 0**,** key**;**

void push**();** void pop**();** void display**();**

int main**()**

**{**

int i**,** size**,** choice**;**

printf**(**"\n\tEnter Number of Elements : "**);** scanf**(**"%d"**, &**size**);**

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

**for(**i **=** 0**;** i **<** size**;** i**++)**

**{**

printf**(**"\t\tEnter Value %d : "**, (**i **+** 1**));** scanf**(**"%d"**, &**stack**[**i**]);**

top**++;**

**}**

getch**(); do**

**{**

system**(**"cls"**);**

printf**(**"\n\n\t----------MENU DRIVEN PROGRAM "**);**

printf**(**"\n\t\t1. Push"**);** printf**(**"\n\t\t2. Pop"**);** printf**(**"\n\t\t3. Display"**);** printf**(**"\n\t\t4. Exit"**);**

printf**(**"\n\n\tEnter Your Choice [1 - 4] : "**);** scanf**(**"%d"**, &**choice**);**

**switch(**choice**)**

**{**

###### case 1:

printf**(**"\n\n\tEnter Value to Push : "**);** scanf**(**"%d"**, &**key**);**

push**();**

getch**(); break;**

###### case 2:

pop**();**

getch**(); break;**

###### case 3:

display**();** getch**();**

###### break; case 4:

printf**(**"\n\n\tExisting the Program. \n"**);**

###### break;

**default:**

printf**(**"\n\n\tnvalid Option. "**);**

**}**

**}while(**choice **!=** 4**);**

###### return 0;

**}**

void push**()**

**{**

**if (**top **>** 9**)**

**{**

printf**(**"\n\n\t Overflow "**); return;**

**}**

stack**[**top**] =** key**;** top**++;**

printf**(**"\n\n\tValue %d is Pushed in Stack."**,** key**); return;**

**}**

void pop**()**

**{**

**if (**top **<** 0**)**

**{**

printf**(**"\n\n\t Underflow "**); return;**

**}**

top**--;**

printf**(**"\n\n\tValue is Popped from Stack."**); return;**

**}**

void display**()**

**{**

**if (**top **<** 0**)**

**{**

printf**(**"\n\n\t Underflow "**); return;**

**}**

int i**;**

printf**(**"\n\n\tThe Elements of Stack are : "**); for(**i **=** 0**;** i **<** top**;** i**++)**

**{**

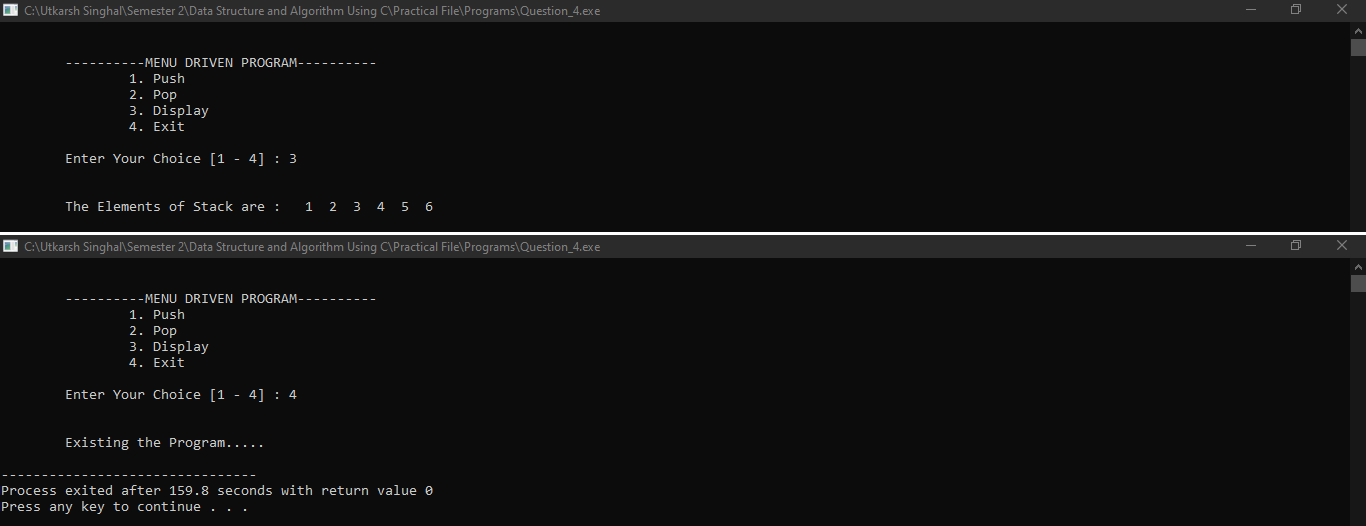
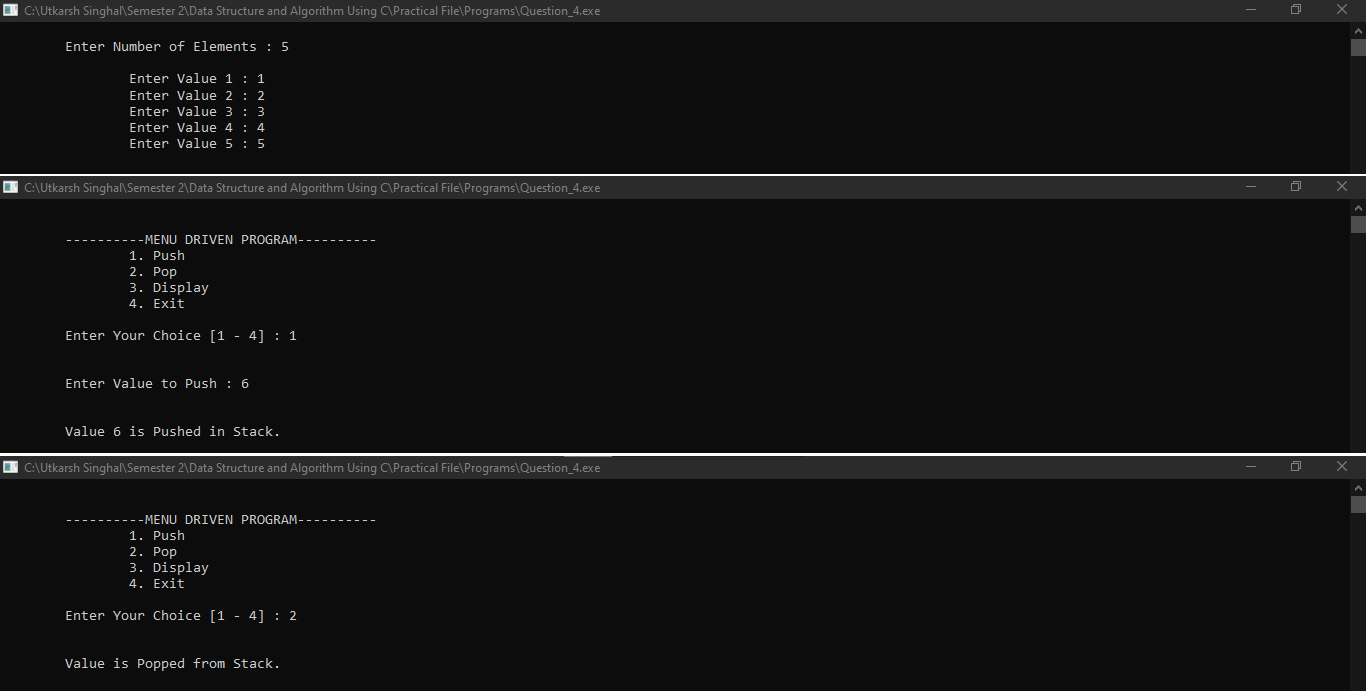
printf**(**" %d"**,** stack**[**i**]);**

**}**

###### return;

**}**

### OUTPUT:



**Code:**

#include<stdio.h>

int queue**[**10**];**

int front **=** 0**,** rear **= -**1**,** count **=** 0**,** key**;**

void insert**();** void delete**();** void display**();**

int main**()**

**{**

int i**,** size**,** choice**;**

printf**(**"\n\tEnter Number of Elements : "**);** scanf**(**"%d"**, &**size**);**

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

**for(**i **=** 0**;** i **<** size**;** i**++)**

**{**

printf**(**"\t\tEnter Value %d : "**, (**i **+** 1**));** scanf**(**"%d"**, &**queue**[**i**]);**

rear **= (**rear **+** 1**) %** 10**;** count**++;**

**}**

getch**(); do**

**{**

system**(**"cls"**);**

printf**(**"\n\n\t----------MENU DRIVEN PROGRAM "**);**

printf**(**"\n\t\t1. Insert"**);** printf**(**"\n\t\t2. Delete"**);** printf**(**"\n\t\t3. Display"**);** printf**(**"\n\t\t4. Exit"**);**

printf**(**"\n\n\tEnter Your Choice [1 - 4] : "**);** scanf**(**"%d"**, &**choice**);**

**switch(**choice**)**

**{**

###### case 1:

printf**(**"\n\n\tEnter Value to Insert : "**);** scanf**(**"%d"**, &**key**);**

insert**();**

getch**(); break;**

###### case 2:

delete**();**

getch**(); break;**

###### case 3:

display**();** getch**();**

###### break; case 4:

printf**(**"\n\n\tExisting the Program. \n"**);**

###### break;

**default:**

printf**(**"\n\n\tnvalid Option. "**);**

**}**

**}while(**choice **!=** 4**);**

###### return 0;

**}**

void insert**()**

**{**

**if (**rear **>=** 9**)**

**{**

printf**(**"\n\n\t Overflow "**); return;**

**}**

rear **= (**rear **+** 1**) %** 10**;** queue**[**rear**] =** key**;** count**++;**

printf**(**"\n\n\tValue %d is Inserted in Queue."**,** key**); return;**

**}**

void delete**()**

**{**

**if (**front **<** 0**)**

**{**

printf**(**"\n\n\t Underflow "**); return;**

**}**

front **= (**front **+** 1**) %** 10**;** count**--;**

printf**(**"\n\n\tValue is Deleted from Queue."**); return;**

**}**

void display**()**

**{**

**if (**front **<** 0**)**

**{**

printf**(**"\n\n\t Underflow "**); return;**

**}**

int i**,** j**;**

printf**(**"\n\n\tThe Elements of Queue are : "**);**

**for(**i **=** front**,** j **=** 0**;** j **<** count**;** j**++,** i **= (**i **+** 1**) %** 10**)**

**{**

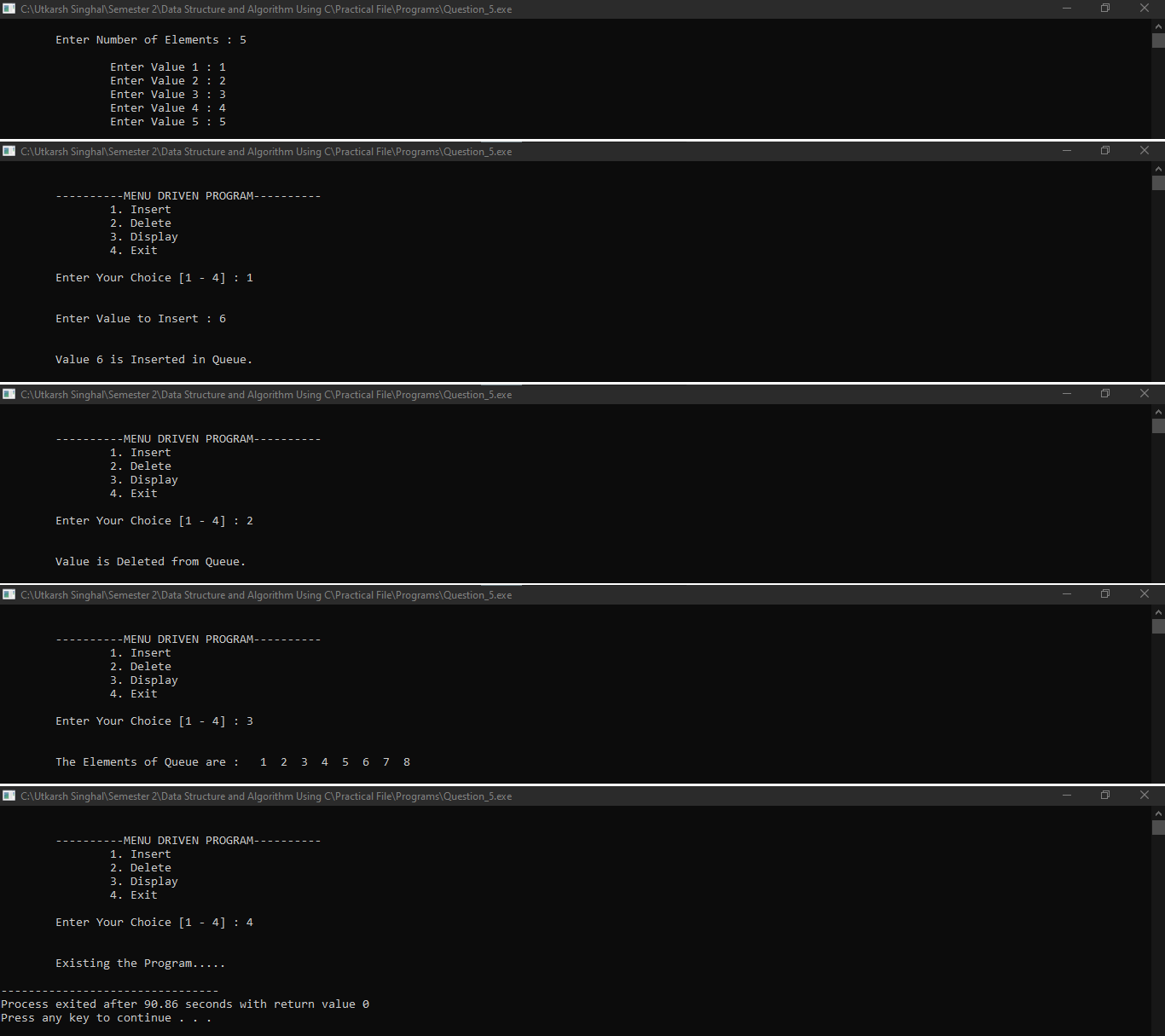
printf**(**" %d"**,** queue**[**i**]);**

**}**

###### return;

**}**

### OUTPUT:



**PRACTICAL : 8**

WAP to implement static And dynamic de-queue.

**Code:**

#include<stdio.h>

int dequeue**[**10**];**

int front **=** 0**,** rear **= -**1**,** count **=** 0**;**

void input**();** void output**();**

void insert\_front**();** void insert\_rear**();** void delete\_front**();** void delete\_rear**();** void display**();**

int main**()**

**{**

int choice**;**

###### do

**{**

system**(**"cls"**);**

printf**(**"\n\n\t\t----------MENU DRIVEN PROGRAM "**);**

printf**(**"\n\t\t\t1. Input Restriced Queue"**);** printf**(**"\n\t\t\t2. Output Restricted Queue"**);** printf**(**"\n\t\t\t3. Exit"**);**

printf**(**"\n\n\tEnter Your Choice [1 - 3] : "**);** scanf**(**"%d"**, &**choice**);**

**switch(**choice**)**

**{**

###### case 1:

input**(); break;**

###### case 2:

output**(); break;**

###### case 3:

printf**(**"\n\n\t\tExisting the Program. \n"**);**

###### break;

**default:**

printf**(**"\n\n\t\tInvalid Option. "**);**

**}**

**}while (**choice **!=** 3**);**

###### return 0;

**}**

void input**()**

**{**

int choice\_input**;**

###### do

**{**

system**(**"cls"**);**

printf**(**"\n\n\t\t----------Input Restriced Queue "**);**

printf**(**"\n\t\t\t1. Insert at Rear"**);** printf**(**"\n\t\t\t2. Delete from Front"**);** printf**(**"\n\t\t\t3. Delete from Rear"**);** printf**(**"\n\t\t\t4. Display"**);** printf**(**"\n\t\t\t5. Exit"**);**

printf**(**"\n\n\tEnter Your Choice [1 - 5] : "**);** scanf**(**"%d"**, &**choice\_input**);**

**switch(**choice\_input**)**

**{**

###### case 1:

insert\_rear**();** getch**(); break;**

###### case 2:

delete\_front**();** getch**(); break;**

###### case 3:

delete\_rear**();** getch**(); break;**

###### case 4:

display**();** getch**(); break;**

###### case 5:

printf**(**"\n\n\t\tReturning to Main Menu. \n"**);**

getch**(); break;**

###### default:

printf**(**"\n\n\t\tInvalid Option. "**);**

**}**

**}while (**choice\_input **!=** 5**);**

###### return;

**}**

void output**()**

**{**

int choice\_output**;**

###### do

**{**

system**(**"cls"**);**

printf**(**"\n\n\t\t----------Output Restriced Queue "**);**

printf**(**"\n\t\t\t1. Insert at Front"**);** printf**(**"\n\t\t\t2. Insert at Rear"**);** printf**(**"\n\t\t\t3. Delete from Front"**);** printf**(**"\n\t\t\t4. Display"**);** printf**(**"\n\t\t\t5. Exit"**);**

printf**(**"\n\n\tEnter Your Choice [1 - 5] : "**);** scanf**(**"%d"**, &**choice\_output**);**

**switch(**choice\_output**)**

**{**

###### case 1:

insert\_front**();** getch**(); break;**

###### case 2:

insert\_rear**();** getch**(); break;**

###### case 3:

delete\_front**();** getch**(); break;**

###### case 4:

display**();** getch**(); break;**

###### case 5:

printf**(**"\n\n\t\tReturning to Main Menu. \n"**);**

getch**(); break;**

###### default:

printf**(**"\n\n\t\tInvalid Option. "**);**

**}**

**}while (**choice\_output **!=** 5**);**

###### return;

**}**

void insert\_front**()**

**{**

**if (**count **>** 9**)**

**{**

printf**(**"\n\n\t\t Overflow "**); return;**

**}**

front **= (**front **+** 9**) %** 10**;**

printf**(**"\n\n\tEnter Value to Insert : "**);** scanf**(**"%d"**, &**dequeue**[**front**]);**

count**++;**

printf**(**"\n\n\tValue %d is Inserted in Queue."**,** dequeue**[**front**]); return;**

**}**

void insert\_rear**()**

**{**

**if (**count **>** 9**)**

**{**

printf**(**"\n\n\t\t Overflow "**); return;**

**}**

rear **= (**rear **+** 1**) %** 10**;**

printf**(**"\n\n\tEnter Value to Insert : "**);** scanf**(**"%d"**, &**dequeue**[**rear**]);**

count**++;**

printf**(**"\n\n\tValue %d is Inserted in Queue."**,** dequeue**[**rear**]); return;**

**}**

void delete\_front**()**

**{**

**if (**count **<** 1**)**

**{**

printf**(**"\n\n\t\t Underflow "**); return;**

**}**

front **= (**front **+** 1**) %** 10**;** count**--;**

printf**(**"\n\n\tValue is Deleted in Queue."**); return;**

**}**

void delete\_rear**()**

**{**

**if (**count **<** 1**)**

**{**

printf**(**"\n\n\t\t Underflow "**); return;**

**}**

rear **= (**rear **+** 9**) %** 10**;** count**--;**

printf**(**"\n\n\tValue is Deleted in Queue."**); return;**

**}**

void display**()**

**{**

**if (**count **<** 1**)**

**{**

printf**(**"\n\n\t\t Underflow"**); return;**

**}**

int i**,** j**;**

printf**(**"\n\n\tThe Elements of Stack are : "**);**

**for(**i **=** front**,** j **=** 0**;** j **<** count**;** j**++,** i **= (**i **+** 1**) %** 10**)**

**{**

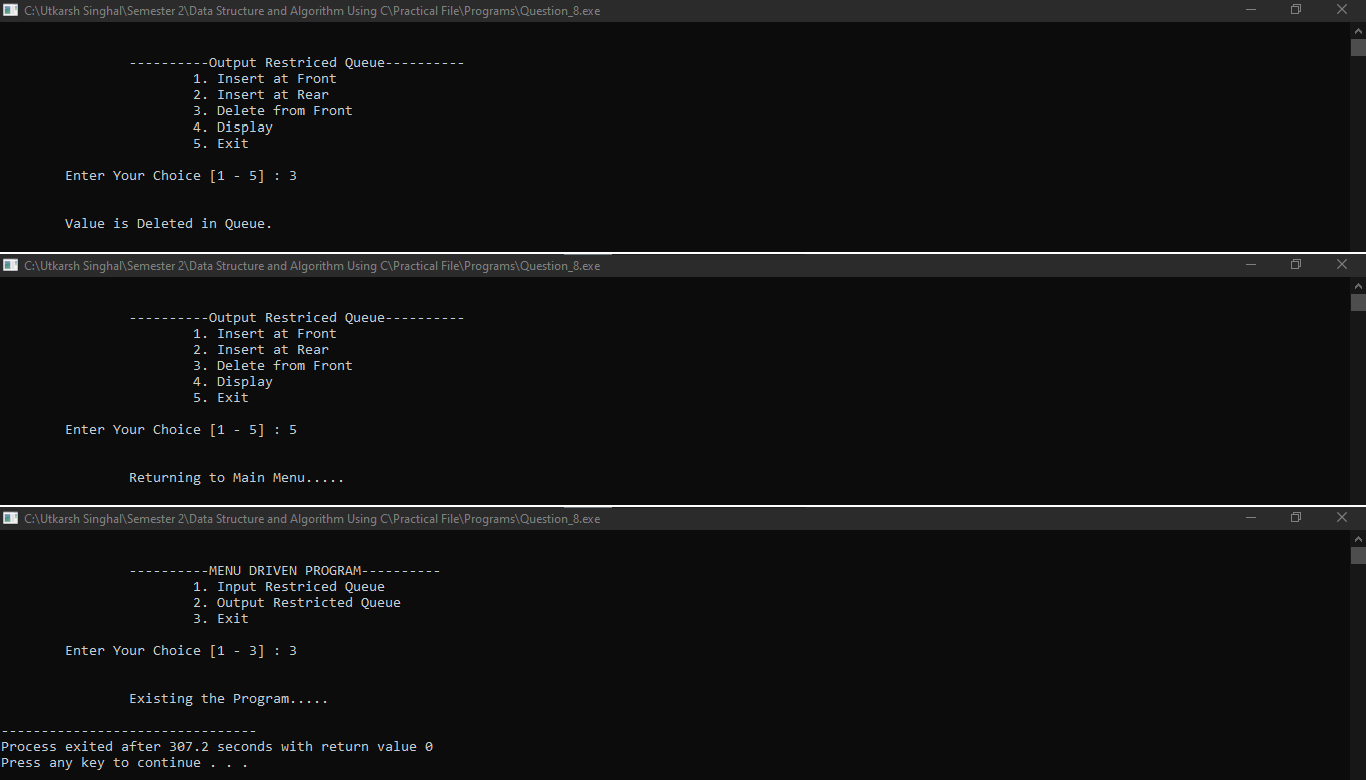
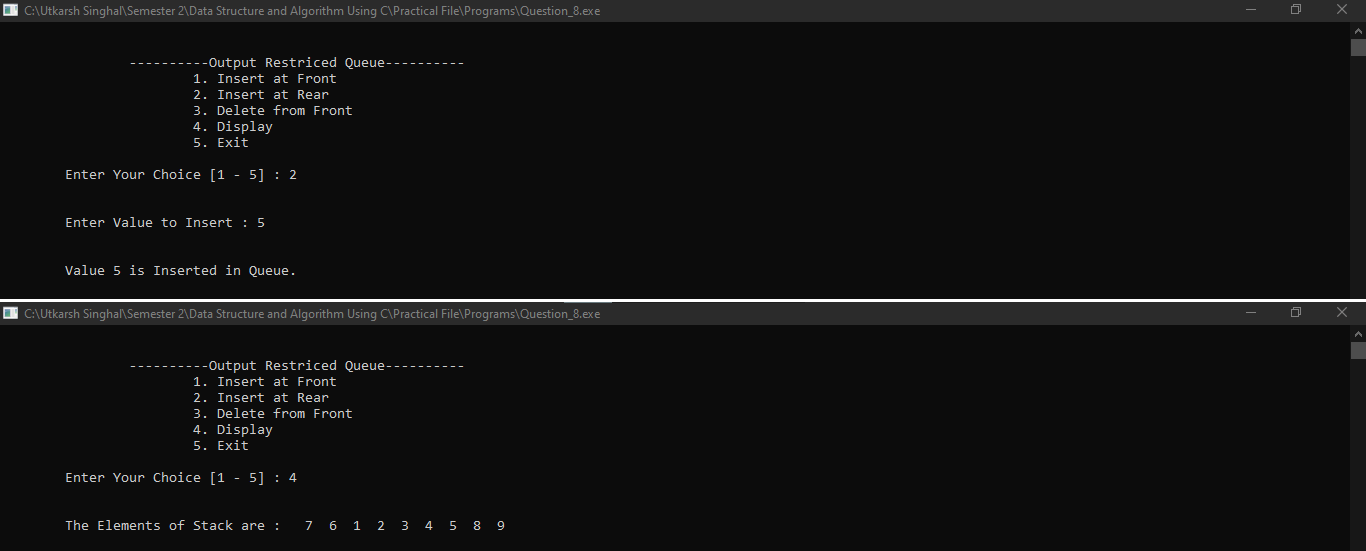
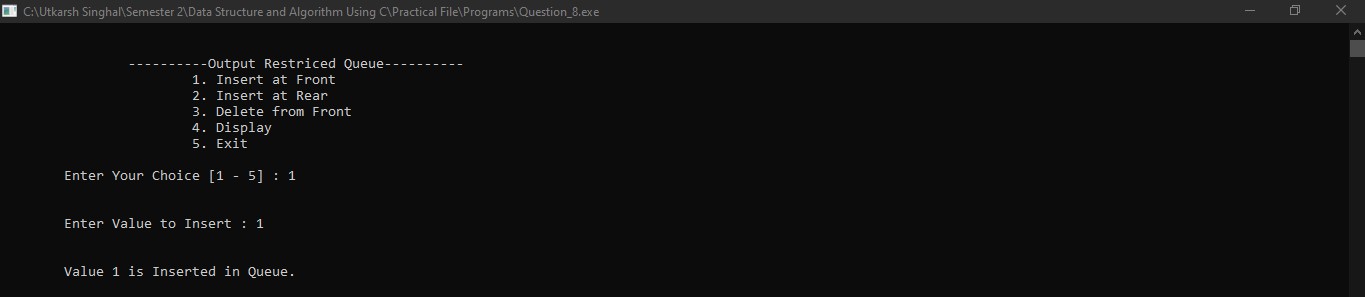
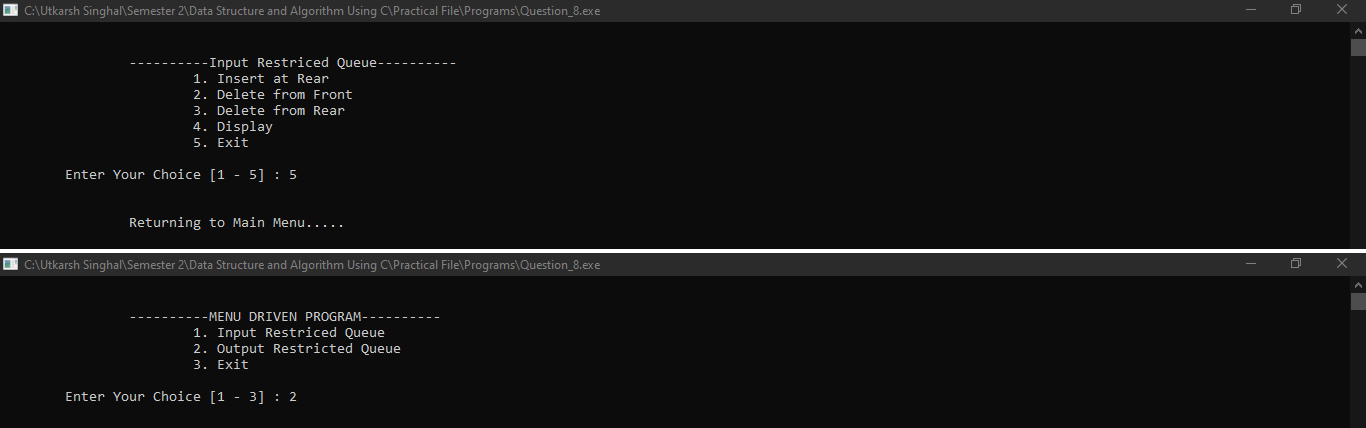
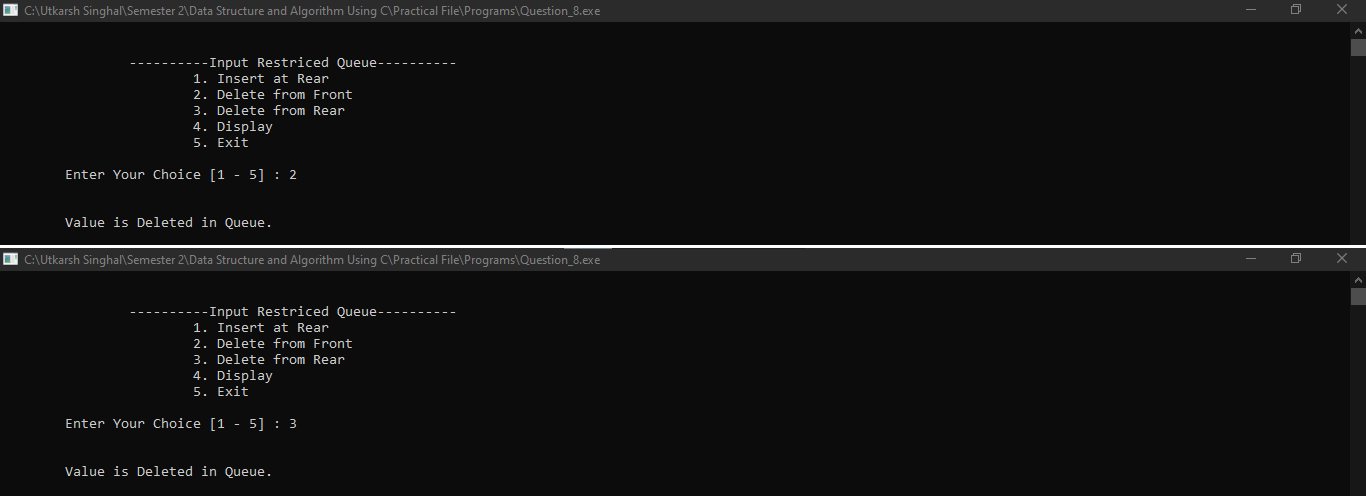
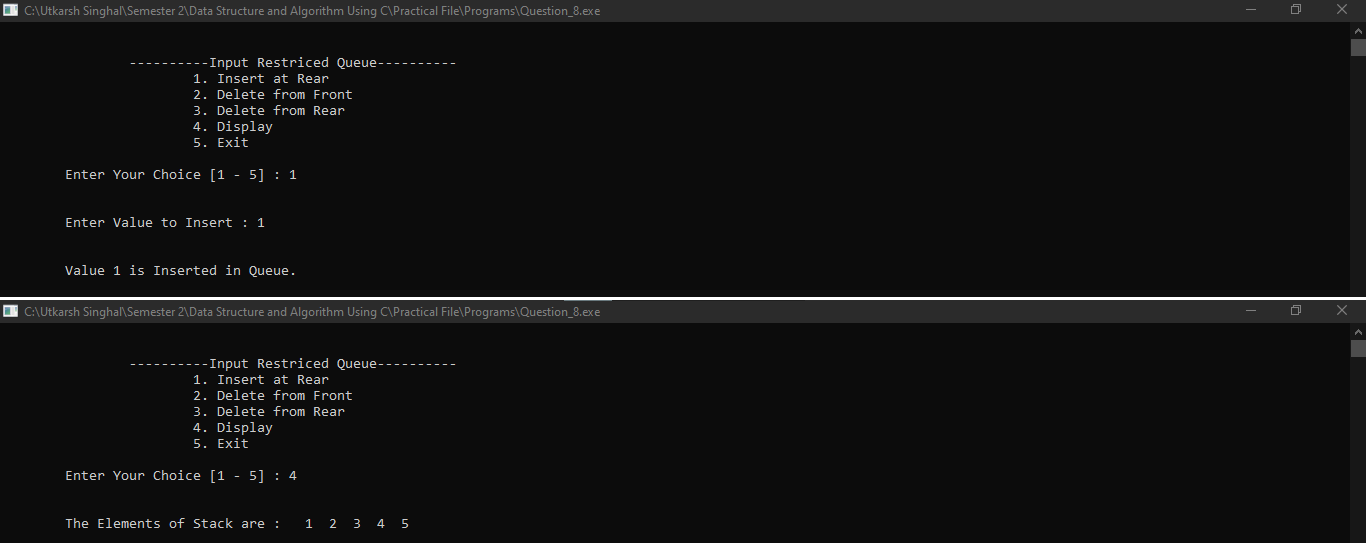
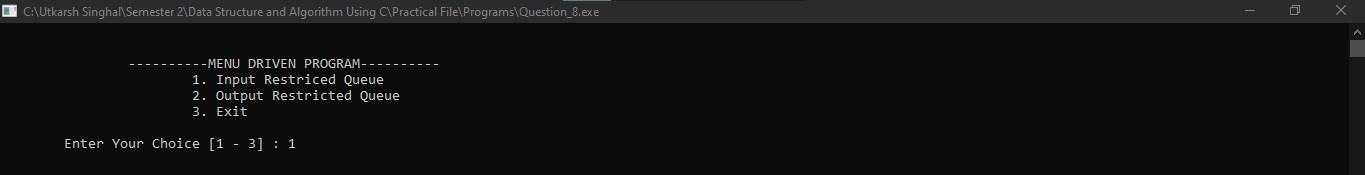
printf**(**" %d"**,** dequeue**[**i**]);**

**}**

###### return;

**}**

### OUTPUT:



**PRACTICAL :18**

Let us assume a Patient's coupon generator for the Doctors’ clinic. The patients are given the coupons on first-come-first-serve basis. After the visit of a patient, patient-ID is kept stack-wise. At the end of the day, the count is generated from the stack. Construct a menu-based program for patients’ coupons generator using an appropriate data structure.

**Code:**

#include<stdio.h>

int queue**[**10**],** stack**[**10**];**

int front **=** 0**,** rear **= -**1**,** top **=** 0**,** key**;**

void insert**();** void delete**();** void push**();**

int main**()**

**{**

int i**,** size**,** choice**;**

printf**(**"\n\tEnter Number of Patients : "**);** scanf**(**"%d"**, &**size**);**

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

**for(**i **=** 0**;** i **<** size**;** i**++)**

**{**

printf**(**"\t\tEnter Patient ID for %d : "**, (**i **+** 1**));** scanf**(**"%d"**, &**queue**[**i**]);**

rear**++;**

**}**

getch**(); do**

**{**

system**(**"cls"**);**

printf**(**"\n\n\t----------MENU DRIVEN PROGRAM "**);**

printf**(**"\n\t\t1. Enter New Patient ID."**);** printf**(**"\n\t\t2. Visiting Complete"**);** printf**(**"\n\t\t3. Show Total Patients"**);** printf**(**"\n\t\t4. Exit"**);**

printf**(**"\n\n\tEnter Your Choice [1 - 4] : "**);** scanf**(**"%d"**, &**choice**);**

**switch(**choice**)**

**{**

###### case 1:

printf**(**"\n\n\tEnter Patient ID to Insert : "**);** scanf**(**"%d"**, &**key**);**

insert**();**

getch**(); break;**

###### case 2:

delete**()**

getch**(); break;**

###### case 3:

printf**(**"\n\n\tTotal Number of Patients : %d"**,** top**);** getch**();**

###### break; case 4:

printf**(**"\n\n\tExisting the Program. \n"**);**

###### break;

**default:**

printf**(**"\n\n\tnvalid Option. "**);**

**}**

**}while(**choice **!=** 4**);**

###### return 0;

**}**

void insert**()**

**{**

**if (**rear **>=** 9**)**

**{**

printf**(**"\n\n\t Overflow "**); return;**

**}**

rear**++;** queue**[**rear**] =** key**;**

printf**(**"\n\n\tPatient ID %d is Inserted in Queue."**,** key**); return;**

**}**

void delete**()**

**{**

**if (**front **>** rear**)**

**{**

printf**(**"\n\n\t Underflow "**); return;**

**}**

push**();** front**++;**

printf**(**"\n\tPatient ID is Deleted from Queue."**); return;**

**}**

void push**()**

**{**

**if (**top **>** 9**)**

**{**

printf**(**"\n\n\t Overflow "**); return;**

**}**

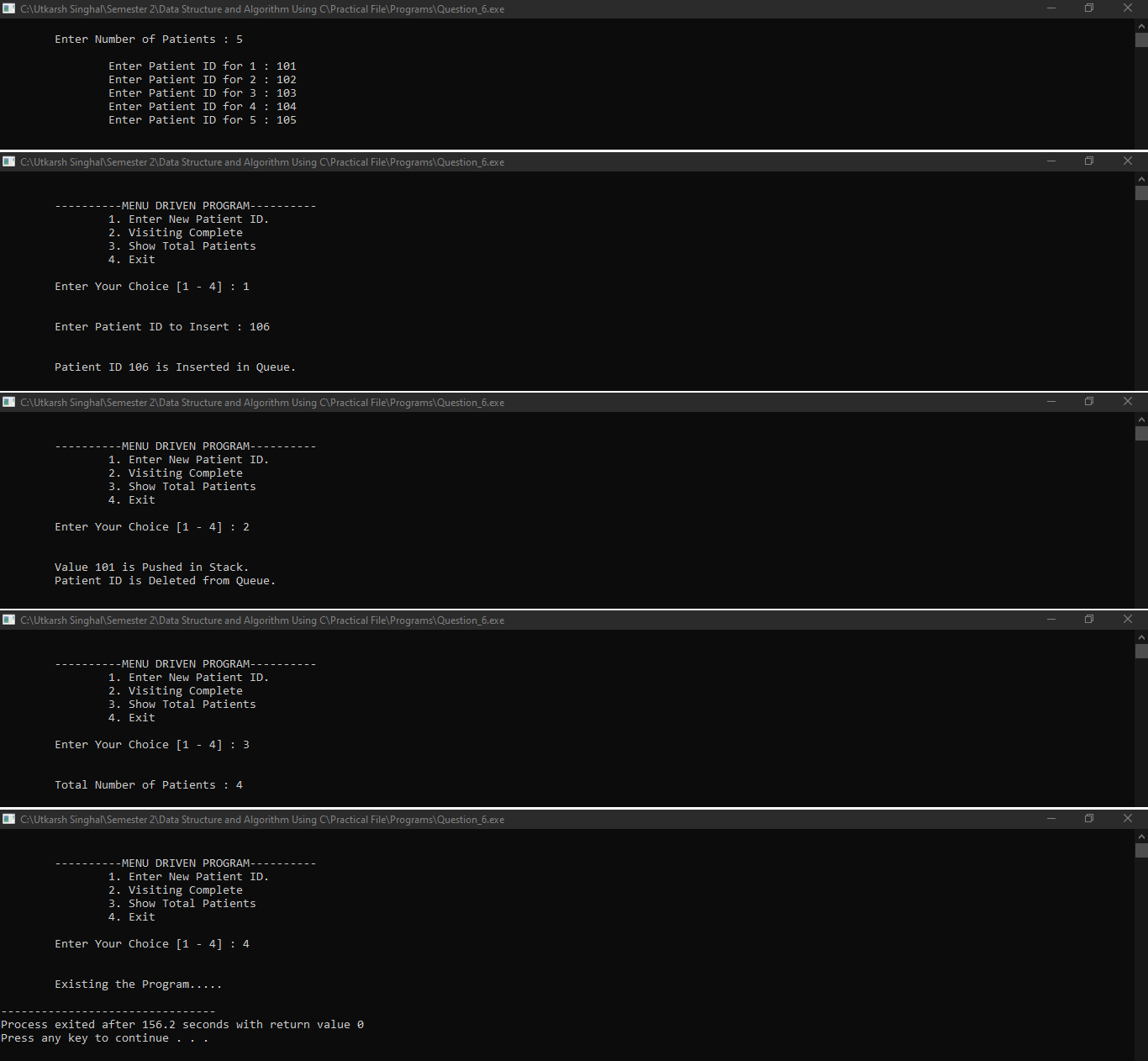
stack**[**top**] =** queue**[**front**];** top**++;**

printf**(**"\n\n\tValue %d is Pushed in Stack."**,** queue**[**front**]);**

###### return;

**}**

### OUTPUT:



**PRACTICAL:7**

WAP to implement a Static and Dynamic Circular Queue.

**Code:**

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

struct Node **{**

int data**;**

struct Node**\*** prev**;** struct Node**\*** next**;**

**};**

struct Node**\*** front **= NULL;** struct Node**\*** rear **= NULL;** struct Node**\*** temp **= NULL;** int count **=** 0**,** new\_data**;**

void insert**(**struct Node**\*\*** head\_Ref**,** struct Node**\*\*** rear\_Ref**);** void delete**();**

void display**();**

int main**()**

**{**

int choice**;**

###### do

**{**

system**(**"cls"**);**

printf**(**"\n\n\t\t----------MENU DRIVEN PROGRAM "**);**

printf**(**"\n\t\t\t1. Insert an Element."**);** printf**(**"\n\t\t\t2. Delete an Element."**);** printf**(**"\n\t\t\t3. Display."**);** printf**(**"\n\t\t\t4. Exit."**);**

printf**(**"\n\n\tEnter Your Choice [1 - 4] : "**);** scanf**(**"%d"**, &**choice**);**

**switch (**choice**)**

**{**

Queue : "**);**

**case** 1**:**

printf**(**"\n\n\t\t\tEnter New Data to Insert in Circular

scanf**(**"%d"**, &**new\_data**);** insert**(&**front**, &**rear**);** getch**();**

###### break;

**case** 2**:**

delete**();**

getch**(); break;**

###### case 3:

display**();** getch**(); break;**

###### case 4:

printf**(**"\n\n\t\tExisting the Program. \n"**);**

###### break;

**default:**

printf**(**"\n\n\t\tInvalid Option. "**);**

getch**(); break;**

**}**

**}while(**choice **!=** 4**);**

###### return 0;

**}**

void insert**(**struct Node**\*\*** head\_Ref**,** struct Node**\*\*** rear\_Ref**)**

**{**

temp **= (**struct Node**\*)** malloc**(sizeof(**struct Node**));** temp **->** data **=** new\_data**;**

temp **->** next **= NULL;**

**if (\***head\_Ref **== NULL)**

**{**

**}**

###### else

**{**

**}**

temp **->** prev **= NULL; (\***head\_Ref**) =** temp**; (\***rear\_Ref**) =** temp**;**

**(\***rear\_Ref**) ->** next **=** temp**;** temp **->** prev **= (\***rear\_Ref**); (\***rear\_Ref**) =** temp**;**

printf**(**"\n\n\t\tElement %d is Added in Circular Queue."**,** new\_data**);** count**++;**

###### return;

**}**

void delete**()**

**{**

**if (**count **==** 0**)**

**{**

printf**(**"\n\n\t\tCircular Queue is Empty."**); return;**

**}**

temp **=** front**;**

front **=** front **->** next**;** front **->** prev **= NULL;** free**(**temp**);**

count**--;**

printf**(**"\n\n\t\tElement is Deleted from Circular Queue."**); return;**

**}**

void display**()**

**{**

temp **=** front**;**

**if (**count **==** 0**)**

**{**

printf**(**"\n\t\tCircular Queue is Empty."**);**

**}**

printf**(**"\n\n\t\tThe Elements of Circular Queue : "**); while (**temp **!= NULL)**

**{**

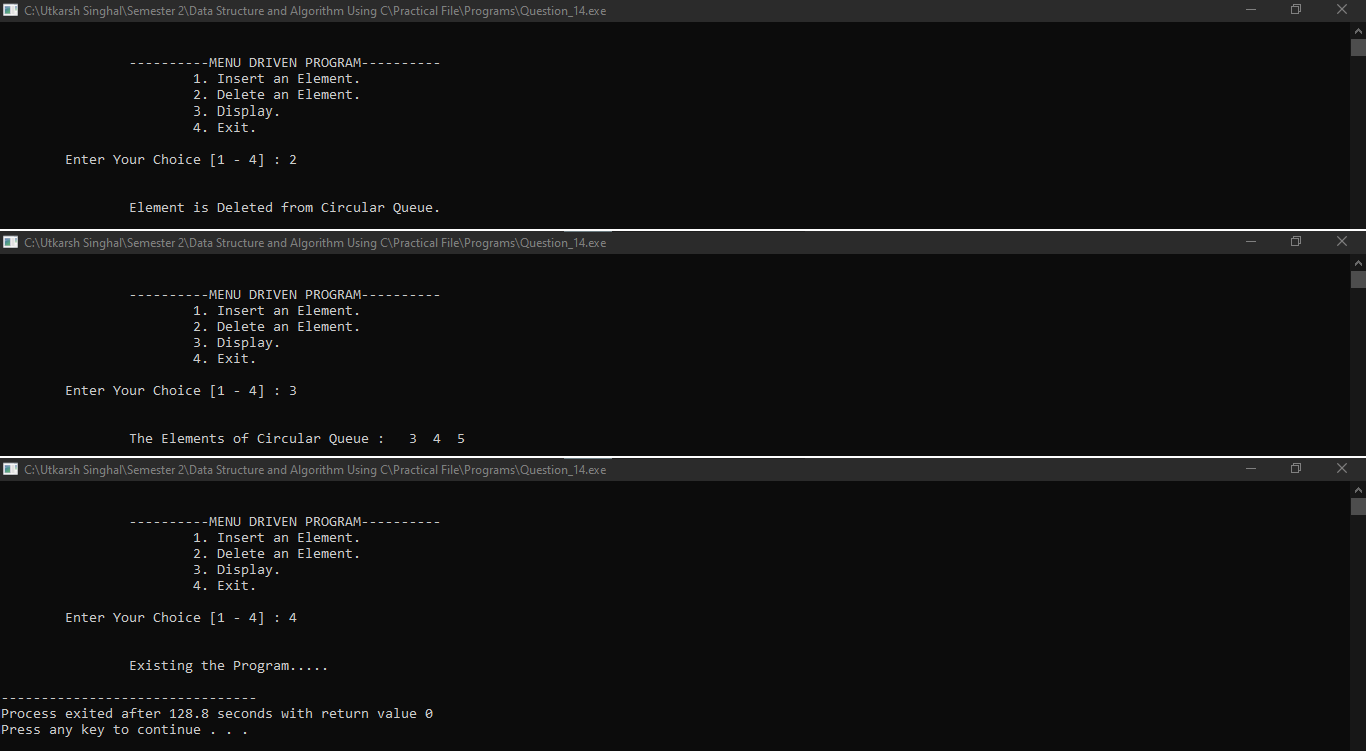
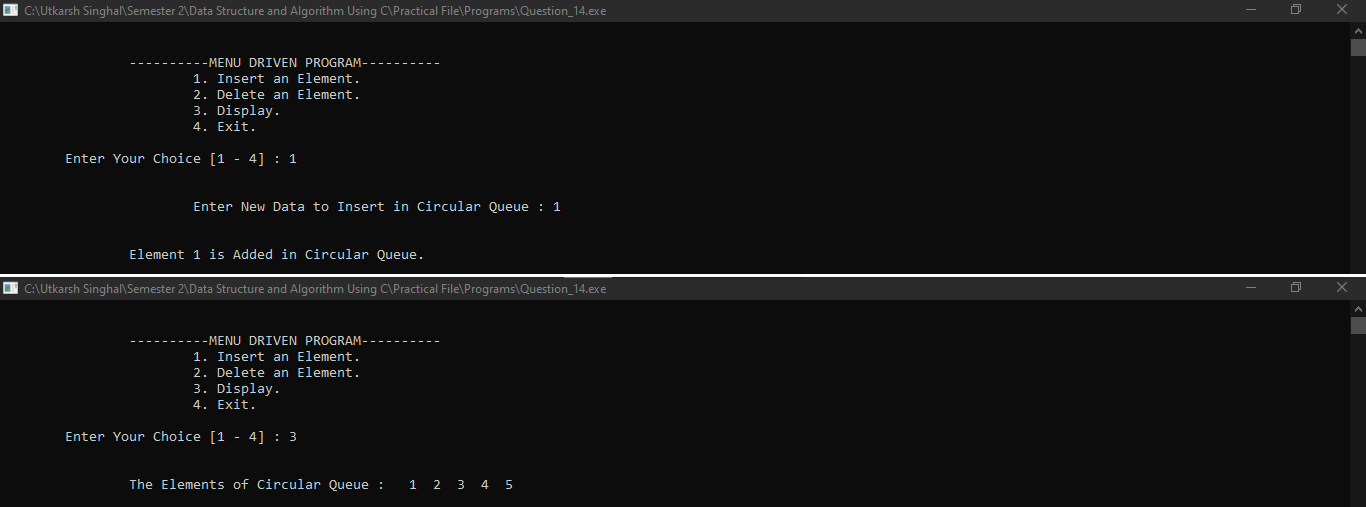
printf**(**" %d"**,** temp **->** data**);** temp **=** temp **->** next**;**

**}**

###### return;

**}**

### OUTPUT:



PRACTICAL:2

WAP to Sort an Array using menu driven**:**

* 1. Bubble Sort
  2. Merge Sort
  3. Insertion Sort
  4. Selection Sort

Code:

#include <stdio.h>

void bubbleSort(int arr[], int n);

void mergeSort(int arr[], int l, int r);

void merge(int arr[], int l, int m, int r);

void insertionSort(int arr[], int n);

void selectionSort(int arr[], int n);

void printArray(int arr[], int n);

int main() {

    int choice, n;

    printf("Enter the number of elements in the array: ");

    scanf("%d", &n);

    int arr[n];

    printf("Enter the elements of the array:\n");

    for (int i = 0; i < n; i++) {

        scanf("%d", &arr[i]);

    }

    printf("\nSelect a sorting algorithm:\n");

    printf("1. Bubble Sort\n");

    printf("2. Merge Sort\n");

    printf("3. Insertion Sort\n");

    printf("4. Selection Sort\n");

    printf("Enter your choice: ");

    scanf("%d", &choice);

    switch(choice) {

        case 1:

            bubbleSort(arr, n);

            break;

        case 2:

            mergeSort(arr, 0, n - 1);

            break;

        case 3:

            insertionSort(arr, n);

            break;

        case 4:

            selectionSort(arr, n);

            break;

        default:

            printf("Invalid choice\n");

    }

    printf("\nSorted array: ");

    printArray(arr, n);

    return 0;

}

void bubbleSort(int arr[], int n) {

    for (int i = 0; i < n - 1; i++) {

        for (int j = 0; j < n - i - 1; j++) {

            if (arr[j] > arr[j + 1]) {

                int temp = arr[j];

                arr[j] = arr[j + 1];

                arr[j + 1] = temp;

            }

        }

    }

}

void mergeSort(int arr[], int l, int r) {

    if (l < r) {

        int m = l + (r - l) / 2;

        mergeSort(arr, l, m);

        mergeSort(arr, m + 1, r);

        merge(arr, l, m, r);

    }

}

void merge(int arr[], int l, int m, int r) {

    int i, j, k;

    int n1 = m - l + 1;

    int n2 = r - m;

    int L[n1], R[n2];

    for (i = 0; i < n1; i++)

        L[i] = arr[l + i];

    for (j = 0; j < n2; j++)

        R[j] = arr[m + 1 + j];

    i = 0;

    j = 0;

    k = l;

    while (i < n1 && j < n2) {

        if (L[i] <= R[j]) {

            arr[k] = L[i];

            i++;

        } else {

            arr[k] = R[j];

            j++;

        }

        k++;

    }

    while (i < n1) {

        arr[k] = L[i];

        i++;

        k++;

    }

    while (j < n2) {

        arr[k] = R[j];

        j++;

        k++;

    }

}

void insertionSort(int arr[], int n) {

    int i, key, j;

    for (i = 1; i < n; i++) {

        key = arr[i];

        j = i - 1;

        while (j >= 0 && arr[j] > key) {

            arr[j + 1] = arr[j];

            j = j - 1;

        }

        arr[j + 1] = key;

    }

}

void selectionSort(int arr[], int n) {

    int i, j, min\_idx;

    for (i = 0; i < n - 1; i++) {

        min\_idx = i;

        for (j = i + 1; j < n; j++)

            if (arr[j] < arr[min\_idx])

                min\_idx = j;

        int temp = arr[min\_idx];

        arr[min\_idx] = arr[i];

        arr[i] = temp;

    }

}

void printArray(int arr[], int n) {

    for (int i = 0; i < n; i++) {

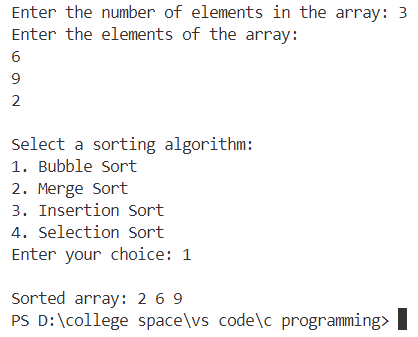
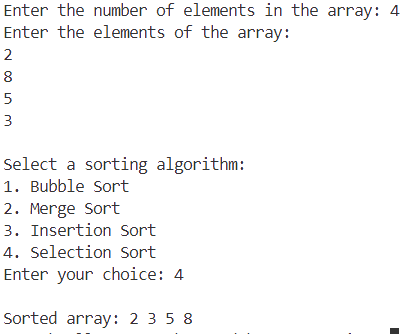
        printf("%d ", arr[i]);

    }

    printf("\n");

}

**Output:**

** **

PRACTICAL:9

Implement recursive algorithms for the following operations on Binary Search Tree:

* 1. Insertion
  2. Searching

**Code:**

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

struct bst **{**

int data**;**

struct bst **\***left**, \***right**;**

**};**

struct bst**\*** root **= NULL;**

int new\_data**,** key**;**

void insert**(**struct bst**\*\*** root**,** int value**);** int search**(**struct bst**\*** root**,** int value**);**

int main**()**

**{**

int choice**;**

###### do

**{**

system**(**"cls"**);**

printf**(**"\n\n\t\t----------MENU DRIVEN PROGRAM "**);**

printf**(**"\n\t\t\t1. Insert a Node"**);** printf**(**"\n\t\t\t2. Search a Node"**);** printf**(**"\n\t\t\t6. Exit"**);**

printf**(**"\n\n\t\tEnter your choice: "**);** scanf**(**"%d"**, &**choice**);**

**switch(**choice**)**

**{**

###### case 1:

printf**(**"\n\t\t\tEnter Data to Insert : "**);** scanf**(**"%d"**, &**new\_data**);**

insert**(&**root**,** new\_data**);**

printf**(**"\n\n\t\tData is Inserted in Binary Search Tree."**);** getch**();**

###### break; case 2:

key**);**

printf**(**"\n\t\t\tEnter Value to Search : "**);** scanf**(**"%d"**, &**key**);**

**if(**search**(**root**,** key**))**

printf**(**"\n\t\t\t%d Found in Binary Search Tree"**,** key**); else**

printf**(**"\n\t\t\t%d Not Found in Binary Search Tree"**,**

getch**(); break;**

###### case 6:

printf**(**"\n\n\t\tExisting the Program. \n"**);**

###### break;

**default:**

printf**(**"\n\n\t\tInvalid Option. "**);**

getch**();**

**}**

**}while(**choice **!=** 6**);**

###### return 0;

**}**

void insert**(**struct bst**\*\*** root**,** int value**)**

**{**

**if(\***root **== NULL)**

**{**

**\***root **= (**struct bst**\*)** malloc**(sizeof(**struct bst**)); (\***root**) ->** data **=** value**;**

**(\***root**) ->** left **= NULL; (\***root**) ->** right **= NULL;**

**}**

**else if (**value **< (\***root**) ->** data**)**

**{**

**}**

###### else

**{**

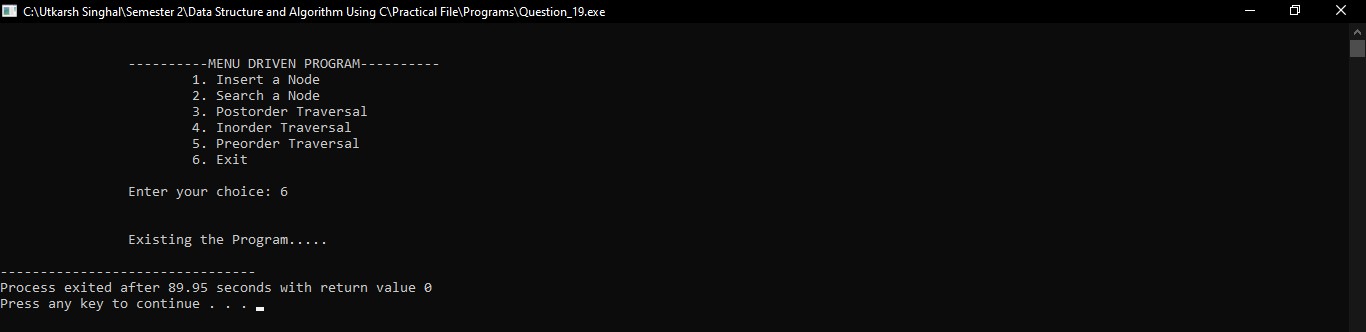
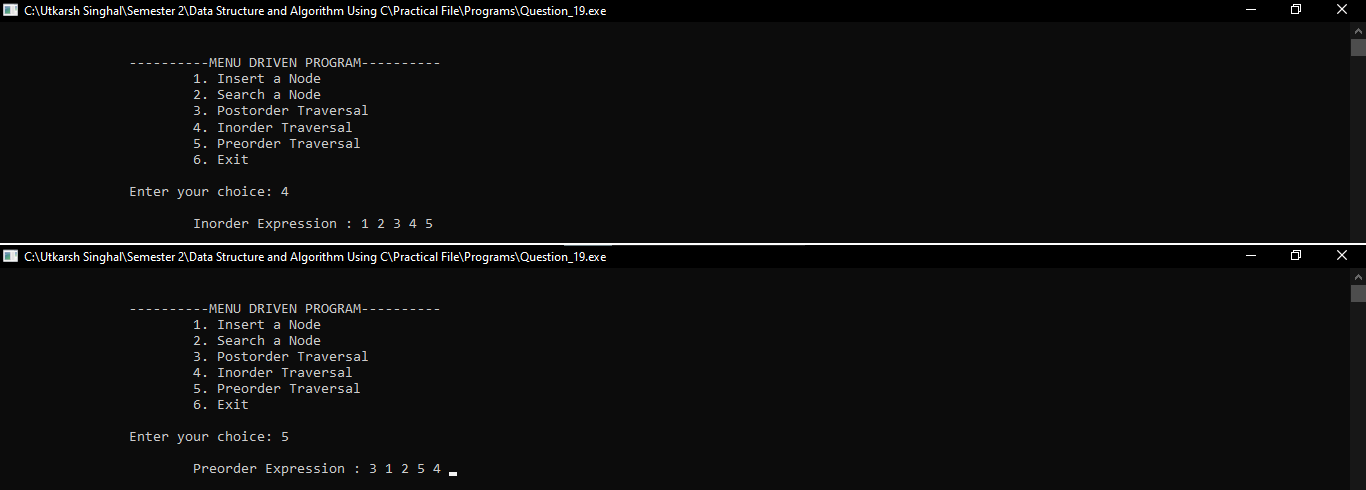
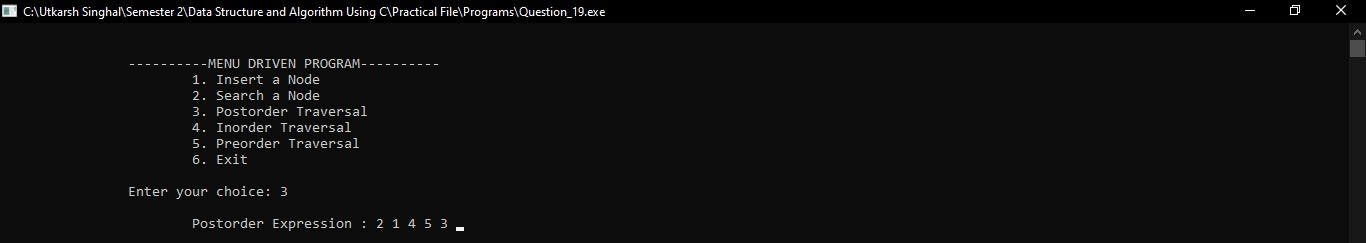
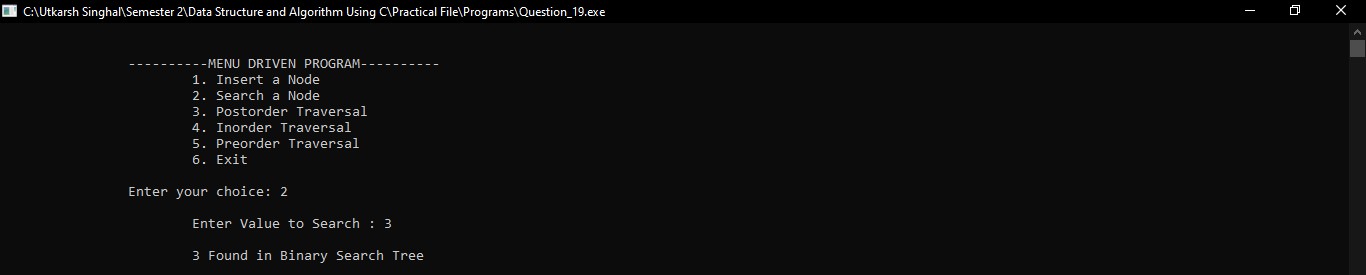
**}**

insert**(&((\***root**) ->** left**),** value**);**

insert**(&((\***root**) ->** right**),** value**);**

###### return;

# OUTPUT:



PRACTICAL 16:

Write a program to convert Infix to equivalent

* 1. Prefix expression
  2. Postfix expression

**Code:**

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

#define MAX\_SIZE 20

char infix**[**MAX\_SIZE**],** prefix**[**MAX\_SIZE**],** postfix**[**MAX\_SIZE**],** stack**[**MAX\_SIZE**];** int stack\_top **= -**1**;**

int isOperator**(**char ch**);** int priority**(**char ch**);** void push**(**char ch**);**

char pop**();**

void infixToPrefix**();** void infixToPostfix**();**

int main**()**

**{**

int choice**;**

###### do

**{**

system**(**"cls"**);**

printf**(**"\n\n\t\t----------MENU DRIVEN PROGRAM "**);**

printf**(**"\n\t\t\t1. Convert Infix to Prefix"**);** printf**(**"\n\t\t\t2. Convert Infix to Postfix"**);** printf**(**"\n\t\t\t3. Exit"**);**

printf**(**"\n\n\t\tEnter Your Choice [1 - 3] : "**);** scanf**(**"%d"**, &**choice**);**

**switch (**choice**)**

**{**

###### case 1:

printf**(**"\n\n\t\tEnter Infix Expression: "**);** scanf**(**"%s"**,** infix**);**

fflush**(**stdin**);** infixToPrefix**();**

printf**(**"\n\n\t\tPrefix Expression: %s"**,** prefix**);** getch**();**

###### break;

**case** 2**:**

printf**(**"\n\n\t\tEnter Infix Expression: "**);** scanf**(**"%s"**,** infix**);**

fflush**(**stdin**);** infixToPostfix**();**

printf**(**"\n\n\t\tPostfix Expression: %s"**,** postfix**);** getch**();**

###### break; case 3:

printf**(**"\n\n\t\tExisting the Program. \n"**);**

###### break;

**default:**

printf**(**"\n\n\t\tInvalid choice!"**);** getch**();**

###### break;

**}**

**}while(**choice **!=** 3**);**

###### return 0;

**}**

void infixToPrefix**()**

**{**

int i**,** j**;**

int len **=** strlen**(**infix**);**

**for (**i **=** len **-** 1**,** j **=** 0**;** i **>=** 0**;** i**--,** j**++)**

**{**

prefix**[**j**] =** infix**[**i**];**

**}**

prefix**[**j**] =** '\0'**;** infixToPostfix**();**

**for (**i **=** 0**,** j **=** strlen**(**postfix**) -** 1**;** j **>=** 0**;** i**++,** j**--)**

**{**

prefix**[**i**] =** postfix**[**j**];**

**}**

prefix**[**i**] =** '\0'**;**

**}**

void infixToPostfix**()**

**{**

int i**,** j**;**

int len **=** strlen**(**infix**);** char ch**;**

**for (**i **=** 0**,** j **=** 0**;** i **<** len**;** i**++)**

**{**

ch **=** infix**[**i**]; if (**isalpha**(**ch**))**

**{**

postfix**[**j**++] =** ch**;**

**}**

**else if (**isOperator**(**ch**))**

**{**

**while (**stack\_top **!= -**1 **&&** priority**(**ch**) <=**

priority**(**stack**[**stack\_top**]))**

**{**

postfix**[**j**++] =** pop**();**

**}**

push**(**ch**);**

**}**

**}**

**while (**stack\_top **!= -**1**)**

**{**

postfix**[**j**++] =** pop**();**

**}**

postfix**[**j**] =** '\0'**;**

**}**

int isOperator**(**char ch**)**

**{**

**if (**ch **==** '+' **||** ch **==** '-' **||** ch **==** '\*' **||** ch **==** '/' **||** ch **==** '%' **||** ch

**==** '^'**)**

**{**

###### return 1;

**}**

###### return 0;

**}**

int priority**(**char ch**)**

**{**

**if (**ch **==** '^'**)**

**{**

###### return 3;

**}**

**else if (**ch **==** '\*' **||** ch **==** '/' **||** ch **==** '%'**)**

**{**

###### return 2;

**}**

**else if (**ch **==** '+' **||** ch **==** '-'**)**

**{**

**}**

###### else

**{**

**}**

**}**

###### return 1;

**return** 0**;**

void push**(**char ch**)**

**{**

**if (**stack\_top **==** MAX\_SIZE **-** 1**)**

**{**

printf**(**"Stack Overflow\n"**); return;**

**}**

stack**[++**stack\_top**] =** ch**;**

**}**

char pop**()**

**{**

**if (**stack\_top **== -**1**)**

**{**

printf**(**"Stack Underflow\n"**); return** '\0'**;**

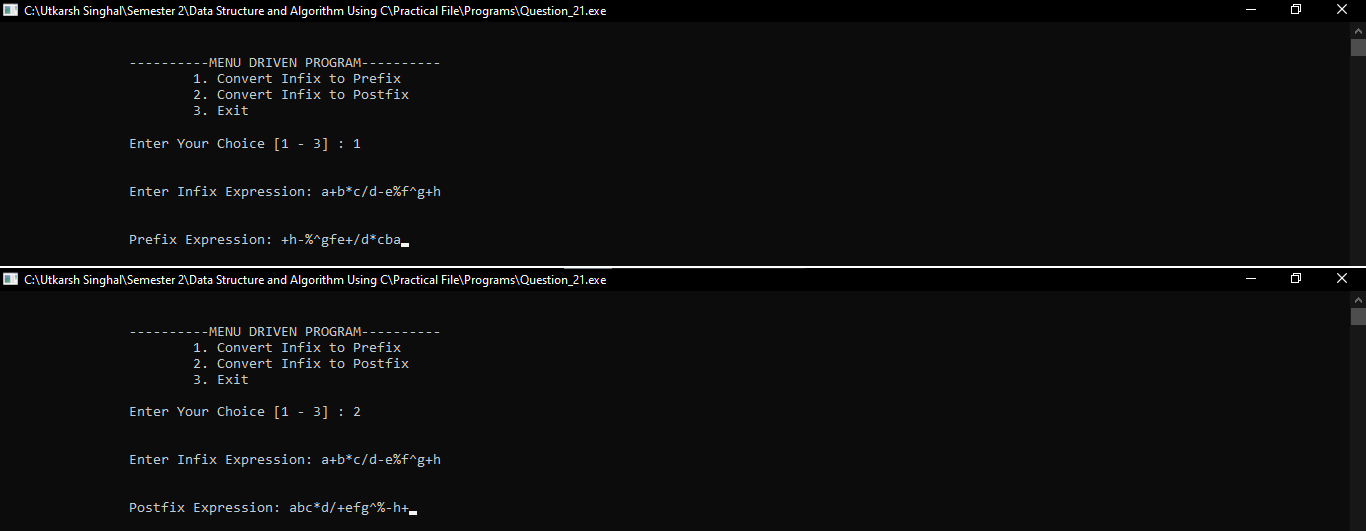
**}**

char exit\_ch **=** stack**[**stack\_top**];** stack\_top**--;**

**return** exit\_ch**;**

**}**

# OUTPUT:



PRACTICAL 17:

Write a program to evaluate

* 1. Prefix expression
  2. Postfix expression

**Code:**

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

int stack**[**20**],** top **= -**1**,** result **=** 0**;** char string**[**20**],** ch**;**

void evaluate**();** int main**()**

**{**

int choice**;**

###### do

**{**

system**(**"cls"**);**

printf**(**"\n\n\t\t----------MENU DRIVEN PROGRAM "**);**

printf**(**"\n\t\t\t1. Evaluate Postfix."**);** printf**(**"\n\t\t\t2. Evaluate Prefix."**);** printf**(**"\n\t\t\t3. Exit."**);**

printf**(**"\n\n\t\tEnter Your Choice [1 - 3] : "**);** scanf**(**"%d"**, &**choice**);**

**switch (**choice**)**

**{**

###### case 1:

printf**(**"\n\t\t\tEnter Expression : "**);** fflush**(**stdin**);**

scanf**(**"%[^\n]s"**,** string**);** evaluate**();**

getch**(); break;**

###### case 2:

printf**(**"\n\t\t\tEnter Expression : "**);** fflush**(**stdin**);**

scanf**(**"%[^\n]s"**,** string**);** strrev**(**string**);** evaluate**();**

getch**(); break;**

###### case 3:

printf**(**"\n\n\t\tExisting the Program. \n"**);**

###### break;

**}**

**}while (**choice **!=** 3**);**

###### return 0;

**}**

void evaluate**()**

**{**

int i**,** top\_1**,** top\_2**;**

**for(**i **=** 0**;** string**[**i**] !=** '\0'**;** i**++)**

**{**

ch **=** string**[**i**]; if(**isalnum**(**ch**))**

**{**

stack**[++**top**] =** ch **-** '0'**;**

**}**

###### else

**{**

**if (**ch **==** ' '**)**

**{**

**}**

###### else

**{**

###### continue;

top\_1 **=** stack**[**top**--];** top\_2 **=** stack**[**top**--];**

###### switch(ch)

**{**

**case** '+'**:**

result **=** top\_2 **+** top\_1**; break;**

**case** '-'**:**

result **=** top\_2 **-** top\_1**; break;**

**case** '\*'**:**

result **=** top\_2 **\*** top\_1**; break;**

**case** '/'**:**

result **=** top\_2 **/** top\_1**; break;**

**}**

stack**[++**top**] =** result**;**

**}**

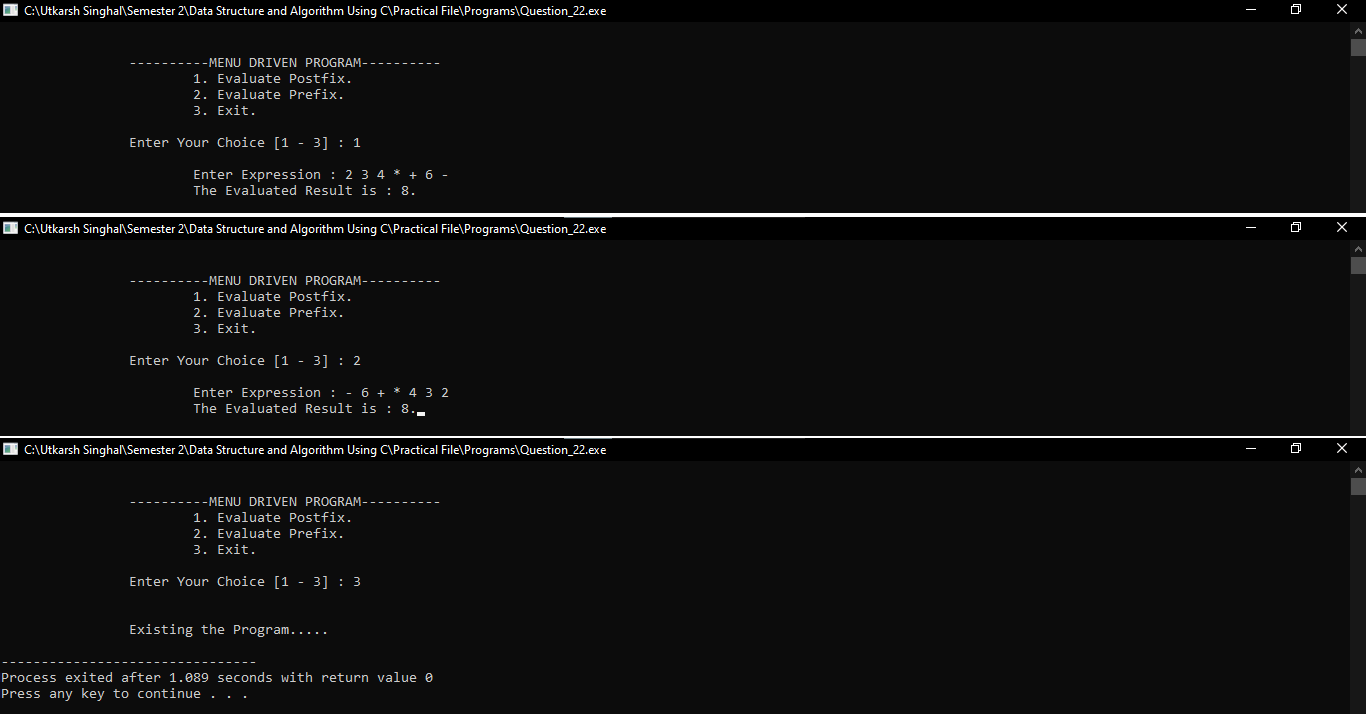
**}**

**}**

printf**(**"\t\t\tThe Evaluated Result is : %d."**,** stack**[**0**]); return;**

**}**

**OUTPUT:**

**mmmbv**

**Practical 10:**

WAP to implement an expression tree. (For Example: (a + b / (c \* d) – e))

Source code:

#include <stdio.h>

#include <stdlib.h>

struct Node {

    char data;

    struct Node\* left;

    struct Node\* right;

};

struct Node\* newNode(char data) {

    struct Node\* node = (struct Node\*)malloc(sizeof(struct Node));

    node->data = data;

    node->left = node->right = NULL;

    return node;

}

// Function to construct the expression tree from a postfix expression

struct Node\* constructExpressionTree(char\* postfix) {

    struct Node\* stack[100];

    int top = -1;

    for (int i = 0; postfix[i] != '\0'; i++) {

        char ch = postfix[i];

        if (isalnum(ch)) {

            struct Node\* temp = newNode(ch);

            stack[++top] = temp;

        } else {

            struct Node\* temp = newNode(ch);

            temp->right = stack[top--];

            temp->left = stack[top--];

            stack[++top] = temp;

        }

    }

    struct Node\* root = stack[top];

    return root;

}

void inorder(struct Node\* root) {

    if (root != NULL) {

        inorder(root->left);

        printf("%c ", root->data);

        inorder(root->right);

    }

}

int evaluate(struct Node\* root) {

    if (root == NULL) {

        return -1;  // Handle invalid expression (empty node)

    }

    if (isalpha(root->data)) {

        return (int)(root->data - 'a' + 1)

    }

    int left\_val = evaluate(root->left);

    int right\_val = evaluate(root->right);

    switch (root->data) {

        case '+': return left\_val + right\_val;

        case '-': return left\_val - right\_val;

        case '\*': return left\_val \* right\_val;

        case '/':

            if (right\_val == 0) {

                printf("Error: Division by zero\n");

                return -1;  // Indicate error

            }

            return left\_val / right\_val;

        default:

            printf("Error: Invalid operator\n");

            return -1;  // Indicate error

    }

}

int main() {

    char postfix[] = "(a+b/(c\*d)-e)";

    struct Node\* root = constructExpressionTree(postfix);

    printf("Inorder traversal (expression): ");

    inorder(root);

    printf("\n");

    int result = evaluate(root);

    if (result != -1) {

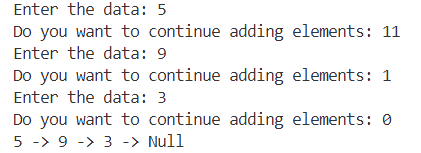
        printf("Evaluation result: %d\n", result);

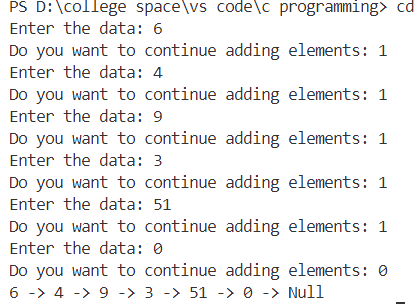
    }

    return 0;

}

**OUTPUT:**

****

****