

LAB-3

1]

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>
#define MAX 100
char st[MAX];
int top = -1;
void push(char st[], char);
char pop(char st[]);
void InfixtoPostfix(char source[], char target[]);
int getpri(char);

void main()
{
    char infix[100], postfix[100];
    printf("\n Enter any infix expression : ");
    gets(infix);
    strcpy(postfix, "");
    InfixtoPostfix(infix, postfix);
    printf("\n The corresponding postfix expression is : ");
    puts(postfix);
}

void InfixtoPostfix(char source[], char target[])
{
    int i = 0, j = 0;
    char temp;
    strcpy(target, "");
    while (source[i] != '\0')
    {
        if (source[i] == '(')
        {
            push(st, source[i]);
            i++;
        }
        else if (source[i] == ')')
        {
            while ((top != -1) && (st[top] != '('))
            {
                target[j] = pop(st);
                j++;
            }
            if (top == -1)
            {
                printf("\n INCORRECT EXPRESSION");
                exit(1);
            }
            temp = pop(st);
            i++;
        }
        else if (isdigit(source[i]) || isalpha(source[i]))
        {
            target[j] = source[i];
            j++;
            i++;
        }
    }
}
```

```

        else if (source[i] == '+' || source[i] == '-' || source[i] == '*'
||
        source[i] == '/' || source[i] == '^' || source[i] == '%')
        {
            while ((top != -1) && (st[top] != '(') && (getpri(st[top]) >
getpri(source[i])))
            {
                target[j] = pop(st);
                j++;
            }
            push(st, source[i]);
            i++;
        }
        else
        {
            printf("\n INCORRECT ELEMENT IN EXPRESSION");
            exit(1);
        }
    }
    while ((top != -1) && (st[top] != '('))
    {
        target[j] = pop(st);
        j++;
    }
    target[j] = '\0';
}
int getpri(char op)
{
    if (op == '^')
        return 2;
    else if (op == '/' || op == '*' || op == '%')
        return 1;
    else if (op == '+' || op == '-')
        return 0;
}
void push(char st[], char val)
{
    if (top == MAX - 1)
        printf("\n STACK OVERFLOW");
    else
    {
        top++;
        st[top] = val;
    }
}
char pop(char st[])
{
    char val = ' ';
    if (top == -1)
        printf("\n STACK UNDERFLOW");
    else
    {
        val = st[top];
        top--;
    }
    return val;
}

```

Output:

```
Select "C:\Users\Sakshi B R\OneDrive\Desktop\ss1.exe"

Enter any infix expression : (A-(B/C+(DEF*F)/G)*H)

The corresponding postfix expression is : ABC/DEF*%G/+H*-

Process returned 0 (0x0)   execution time : 35.246 s
Press any key to continue.
```

```
25 void infixToPostfix(char source[], char target[])
26 {
27     int i = 0, j = 0;
28     char temp;
29     strcpy(target, "");
30     while (source[i] != '\0')
```

Logs & others

```

2] #include <stdio.h>
#define N 5

int q[N];
int front = -1, rear = -1;
void insert(int);
int delete();
void display();
void main()
{
    int n, choice;
    do
    {
        printf("\n1.Insert\n2.Delete\n3.Display\n4.Exit\n");
        printf("Enter your option : \n");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                printf("Enter the number to be inserted in the queue : \n");
                scanf("%d", &n);
                insert(n);
                break;
            case 2:
                n = delete ();
                if (n != -1)
                    printf("\n The number deleted is : %d\n", n);
                break;
            case 3:
                display();
                break;
            case 4:
                exit(0);
                break;
            default:
                printf("Invalid option\n");
                exit(0);
                break;
        }
    } while (choice != 4);
}

void insert(int num)
{
    if ((front == 0 && rear == N - 1) || (rear == (front - 1)))
        printf("\n OVERFLOW");
    else if (front == -1 && rear == -1)
    {
        front = rear = 0;
        q[rear] = num;
    }
    else if (rear == N - 1 && front != 0)
    {
        rear = 0;
        q[rear] = num;
    }
    else
    {

```

```

        rear++;
        q[rear] = num;
    }
}
int delete()
{
    int val;
    if (front == -1 && rear == -1)
    {
        printf("\n UNDERFLOW");
        return -1;
    }
    val = q[front];
    if (front == rear)
        front = rear = -1;
    else
    {
        if (front == N - 1)
            front = 0;
        else
            front++;
    }
    return val;
}
void display()
{
    int i;
    printf("\n");
    if (front == -1 && rear == -1)
        printf("\n QUEUE IS EMPTY");
    else
    {
        if (front < rear)
        {
            for (i = front; i <= rear; i++)
                printf("\t %d", q[i]);
        }
        else
        {
            for (i = front; i < N; i++)
                printf("\t %d", q[i]);
            for (i = 0; i <= rear; i++)
                printf("\t %d", q[i]);
        }
    }
}
}

```

OUTPUT:

```
1.Insert
2.Delete
3.Display
4.Exit
Enter your option :
1
Enter the number to be inserted in the queue :
2
```

```
1.Insert
2.Delete
3.Display
4.Exit
Enter your option :
1
Enter the number to be inserted in the queue :
3
```

```
1.Insert
2.Delete
3.Display
4.Exit
Enter your option :
1
Enter the number to be inserted in the queue :
4
```

```
1.Insert
2.Delete
3.Display
4.Exit
Enter your option :
3
```

```
          2      3      4
1.Insert
2.Delete
3.Display
4.Exit
Enter your option :
2
```

The number deleted is : 2

```
1.Insert
2.Delete
3.Display
4.Exit
```

```

3] #include <stdio.h>
#define N 5

int q[N];
int front = -1, rear = -1;
void insert(int);
int delete();
void display();
void main()
{
    int n, choice;
    do
    {
        printf("\n1.Insert\n2.Delete\n3.Display\n4.Exit\n");
        printf("Enter your option : \n");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                printf("Enter the number to be inserted in the queue : \n");
                scanf("%d", &n);
                insert(n);
                break;
            case 2:
                n = delete ();
                if (n != -1)
                    printf("\n The number deleted is : %d\n", n);
                break;
            case 3:
                display();
                break;
            case 4:
                exit(0);
                break;
            default:
                printf("Invalid option\n");
                exit(0);
                break;
        }
    } while (choice != 4);
}

void insert(int num)
{
    if (rear == N - 1)
        printf("\n OVERFLOW");
    else if (front == -1 && rear == -1)
        front = rear = 0;
    else
        rear++;
    q[rear] = num;
}

int delete()
{
    int val;
    if (front == -1 || front > rear)
    {

```

```

        printf("\n UNDERFLOW");
        return -1;
    }
    else
    {
        val = q[front];
        front++;
        if (front > rear)
            front = rear = -1;
        return val;
    }
}

void display()
{
    int i;
    printf("\n");
    if (front == -1 || front > rear)
        printf("\n QUEUE IS EMPTY");
    else
    {
        for (i = front; i <= rear; i++)
            printf("\t %d", q[i]);
    }
}

```

Output:


```
1.Insert
2.Delete
3.Display
4.Exit
Enter your option :
1
Enter the number to be inserted in the queue :
2

1.Insert
2.Delete
3.Display
4.Exit
Enter your option :
2

    The number deleted is : 2

1.Insert
2.Delete
3.Display
4.Exit
Enter your option :
3

    QUEUE IS EMPTY
1.Insert
2.Delete
3.Display
4.Exit
Enter your option :
1
Enter the number to be inserted in the queue :
3

1.Insert
2.Delete
3.Display
4.Exit
Enter your option :
1
Enter the number to be inserted in the queue :
5

1.Insert
2.Delete
3.Display
```