PAGE NO.:
DATE: / /

Lab - 2

2. Write a program to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators +, -, *, / and ^.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define max 10
int top=-1;
int getPriority(char op);
void infixtopostfix(char source[],
                    char target[]);
int Priority(char);
void push(char st[], char val);
char pop(char st[]);
int 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);
  return 0;
}
```

PAGE NO.:
DATE: / /

```
void infixtopostfix(char source[],
                              char target[])
{
    int i=0, j=0;
    char st[max], 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]=='%')
    {
        while((top!=-1)&& (st[top] != '(')
        && (getPriority (st [top]))>getPriority(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 getPriority (char op)
{
    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='\0';
    if(top==-1)
        printf("\nSTACK UNDERFLOW");
    else
    {
        val=st[top];
        top--;
    }
    return val;
}
```

Output:

```
Enter any infix expression: (A*(B*(((C+D)+B)
                                          * C)))
The corresponding postfix expression
is: ABCD+B+C***
```

DATE: / /

Lab 3

WAP to simulate the working of queue of integers using an array. Provide following operations.
a) Insert b) Delete c) Display
The program should print appropriate messages for queue empty and queue overflow conditions.

```
#include <stdio.h>
#include <conio.h>
#define max 10
int queue [max];
int front = -1, rear = -1;
void insert (void);
int delete element (void);
int peek(void);
void display (void);
int main()
 {
    int option, val;
    do
    {
     printf ("\n*** MAIN MENU ***");
     printf ("\n 1. Insert an element");
     printf("\n 2. Delete an element");
     printf("\n 3. Peek");
     printf(" \n 4. Display the queue");
     printf(" \n 5. Exit");
     printf (" \nEnter your option");
     scanf("%d", &option);
```

DATE: / /

```
switch (option)
{
 case 1:
        insert();
        break;
 case 2:
        val=delete_element();
        if(val!=-1)
         printf("\n The number
                deleted is : %d", val);
        break;
 case 3:
        val = peek();
        if (val!=-1)
         printf("\n The first value
             in queue is: %d", val);
        break;
 case 4:
        display();
        break;
 }
}while (option!=5);
getch();
return 0;
}
void insert()
{
 int num;
 printf("\nEnter the number to be
       inserted in the queue:");
 scanf(" %d", %num);
 if (rear=max-1)
```

```
    printf("\n Overflow");
else if(front == -1 && rear == -1)
  front = rear = 0;
else
   rear++;
   queue [rear] = num;
}
int delete_element()
{
  int val;
  if (front == -1 || front > rear)
  {
    printf("Underflow");
    return -1;
  }
  else
  {
   val = queue[front];
   front ++;
   if (front > rear)
     front = rear = -1;
   return val;
  }
}
int peek()
{
   if(front == -1 || front > rear)
   {
      printf("\n Queue is Empty");
      return -1;
   }
  else
    return queue[front];
}
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", queue[i]);
  }
}
```

Output:

```
*** MAIN MENU***
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option:1

Enter the number to be inserted
in the queue :2
*** MAIN MENU***
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
```

```
    5. Exit
Enter your option :1
 Enter the number to be inserted
 in the queue :1

***MAIN MENU***
  1. Insert an element
  2. Delete an element
  3. Peek
  4. Display the queue
  5. Exit
Enter your option :3
· The first value in queue is:2
```

2) WAP to simulate the working of a circular queue of integers using an array. Provide the following Operations
ⓐ Insert ⓑ Delete ⓒ Display
The program should print appropriate messages for queue empty and queue overflow conditions.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define max 10
int queue[max];
int front = -1 , rear=-1;
void insert (void);
int delete element (void);
int peak(void);
void display (void);
int main()
{
    int option, val;
    do
    {
     printf("\n ** MAIN MENU**");
     printf ("\n 1. Insert an element");
     printf ("\n 2. Delete an element");
     printf ("\n 3. Peek");
     printf ("\n4. Display the queue");
     printf ("\n 5. Exit");
     printf ("\nEnter your option :");
     scanf ("%d",4option);
     switch( option)
     {
      case 1: insert ();
              break;
      case 2: val = delete element();
             if (val!=-1)
                printf("\n The no. deleted
                          is : %d", val);
             break;
case 3: val=peek();
        if (val!=-1)
          printf("\n The first
          value in queue is:%d",val);
        break;
case 4: display();
        break;
 }
}while(option!=5);
getch();
return 0;
}
void insert()
{
 int num;
 printf("\n Enter no to be inserted
          in the queue:");
 scanf("%d",&num);
 if (front==0 && rear==max-1)
   printf("\n Overflow");
 else if (front==-1 && rear==-1)
 {
    front=rear=0;
    queue[rear]=num;
 }
else if (rear==max-1 && front!=0)
{
  rear=0.
  queue [rear]=num;
}
else{
  rear++;
```

```
queue [rear] = num;
}
}
int delete element ()
{
int val;
if (front = = -1 && rear == -1)
{
    printf ("\n Underflow");
    return -1;
}
val = queue [front];
if (front = = rear)
    front = rear = -1;
else {
  if (front == max-1)
      front = 0;
  else
      front ++;
}
int peek ()
{
 if (front == -1 && rear == -1)
 {
   printf ("\n Queue is Empty");
   return -1;
 }
else
  return queue [front];
}
```

```
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", queue [i]);
    }
   else{
    for (i=front ; i<max; i++)
      printf("\t %d", queue [i]);
    for (i=0; i<rear; i++)
      printf("\t %d", queue [i]);
   }
 }
}
```

Output:

```
** MAIN MENU **
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
 Enter your option : 3

Queue is Empty.
```

```
** MAIN MENU**
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. Exit
Enter your option : 1

Enter no. to be inserted in the
queue : 2.
```