```c
#include <stdio.h>
#include <ctype.h>
#define SIZE 50

char stack[SIZE];
int top=-1;
push(char elem)
{
    stack[++top]=elem;
}
char pop()
{
    return(stack[top--]);
}
int pr(char symbol)
{
    if(symbol=='^')
    {
        return(3);
    }
    else if(symbol =='*'|| symbol == '/')
    {
        return(2);
    }
    else if(symbol =='+'|| symbol =='-')
    {
        return(1);
    }
    else
    {
        return(0);
    }
}
void main()
{
    char infix[50],postfix[50],ch,elem;
    int i=0,k=0;
    printf("enter the infix expression:");
    scanf("%s",infix);
    push('#');
    while((ch=infix[i++]) != '\0')
    {
        if( ch =='(') push(ch);
        else
            if(isalnum(ch))postfix[k++]=ch;
        else
            if(ch==')')
        {
            while( stack[top] !='(')
            postfix[k++]=pop();
```

```
    }
    else
    {
        return(0);
    }
}
void main()
{
    char infix[50],postfix[50],ch,elem;
    int i=0,k=0;
    printf("enter the infix expression:");
    scanf("%s",infix);
    push('#');
    while((ch=infix[i++]) != '\0')
    {
        if( ch =='(') push(ch);
        else
            if(isalnum(ch))postfix[k++]=ch;
        else
            if(ch==')')
        {
            while( stack[top] !='(')
            postfix[k++]=pop();
            elem=pop();
        }
        else
        {
            while(pr(stack[top])>=pr(ch))
                postfix[k++]=pop();
            push(ch);
        }
    }
    while(stack[top] != '#')
        postfix[k++]=pop();
    postfix[k]='\0';
    printf("\n Postfix Expression = %s\n",postfix);
}
```

```
enter the infix expression:(8*5-(12/6^3)*4)*3

 Postfix Expression = 85*1263^/4*-3*

Process returned 38 (0x26)   execution time : 100.563 s
Press any key to continue.
```

```c
#include<stdio.h>
int stack[20];
int top=-1;
void push(int x)
{
    stack[++top]=x;
}
int pop()
{
    return stack[top--];
}
int main()
{
    char exp[20];
    char *e;
    int n1,n2,n3,num;
    printf("Enter experssion ::");
    scanf("%s",exp);
    e=exp;

while(*e !='\0')
{
    if(isdigit(*e))
    {
        num=*e -48;
        push(num);
    }
    else
    {
        n1=pop();
        n2=pop();
        switch(*e)
        {
        case '+':
            {
                n3=n1+n2;
                break;
            }
        case '-':
            {
                n3=n2-n1;
                break;
            }
        case '*':
            {
                n3=n2*n1;
                break;
            }
        case '/':
            {
```

```
27      }
28      else
29      {
30          n1=pop();
31          n2=pop();
32          switch(*e)
33          {
34          case '+':
35              {
36                  n3=n1+n2;
37                  break;
38              }
39          case '-':
40              {
41                  n3=n2-n1;
42                  break;
43              }
44          case '*':
45              {
46                  n3=n2*n1;
47                  break;
48              }
49          case '/':
50              {
51                  n3=n2/n1;
52                  break;
53              }
54          }
55          push(n3);
56      }
57      e++;
58  }
59  printf("the result of expression %s= %d",exp,pop());
60  }
61
```

```
Enter experssion ::12*34*+5-
the result of expression 12*34*+5-= 9
Process returned 0 (0x0)   execution time : 37.234 s
Press any key to continue.
```

```c
#include<stdio.h>
#define MAX 50
int queue_array[MAX];
int rear=-1;
int front=-1;
display()
{
  int i;
  if(front==-1)
    printf("queue is empty\n");
else
{
  printf("queue is :\n");
  for(i=front;i<=rear;i++)
    printf("%d",queue_array[i]);
  printf("\n");
}
}
main()
{
  int choice;
  while(1)
  {
    printf("1.insert\n");
    printf("2.delete\n");
    printf("3.display\n");
    printf("4.exit\n");
    printf("enter your choice:");
    scanf("%d",&choice);
    switch(choice)
    {
      case 1:
      insert();
      break;
      case 2:
      delete();
      break;
      case 3:
      display();
      break;
      case 4:
      exit(1);
      break;
      default:
      printf("invalid choice\n");
    }
  }
}
insert()
{
```

```
}
insert()
{
  int add_item;
  if(rear==MAX-1)
    printf("queue overflow\n");
  else
  {
    if(front==-1)
    front=0;
    printf("insert the element in the queue:");
    scanf("%d",&add_item);
    rear+=1;
    queue_array[rear]=add_item;
   }
}
delete()
{
  if(front==-1 || front>rear)
  {
    printf("queue underflow\n");
    return;
  }
  else
  {
    printf("deleted element is : %d\n",queue_array[front]);
    front+=1;
  }
}
```

```
1.insert
2.delete
3.desplay
4.exit
enter your choice:1
insert the element in the queue:15
1.insert
2.delete
3.desplay
4.exit
enter your choice:2
deleted element is : 15
1.insert
2.delete
3.desplay
4.exit
```

LaB - 03 :—

Date: 28/11/23
Page:

1. Write a pgm to convert a given valid pareth -esized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus) - (minus), * (multiply), / (divide) and ^ (power).

→
```
# include <stdio.h>
# include <ctype.h>
# define size 50

char stack [size];
int top = -1;
push (char elem)
{
   stack [++top] = elem;
}
char pop ()
{
   return (stack [top--]);
}
int pr (char symbol)
{
   if (symbol == '^')
   {
      return (3);
   }
   else if (symbol == '*' || symbol == '/')
   {
      return (1);
   }
   else
```

```
    {
      return(1);
    }
    else
    {
      return (0);
    }
}
void main()
  {
    char infix[50], postfix[50,], ch, elem;
    int i=0, k=0;
    printf("Enter the infix expression :");
    scanf("%s" infix);
    push('#');
    while((ch = infix[i++] != '\0')
    {
    if (ch == '(') push (ch);
     else
        if (isalnum (ch) postfix [k++] = ch;
        else
            if (ch == ')')
    {
    while (stack [top] != '(')
      postfix [k++] = pop();
    }     elem = pop();
    else
    {
      while (pr (stack [top]) >= pr(ch))
        postfix [k++] = pop();
        push (ch);
}
```

```
}
while (stack[top] != '#')
    postfix[k++] = pop();
postfix[k] = '\0';
printf("\n Postfix Expression = %s\n",
                    postfix);
}
```

→ output :

Enter infix expression (8*5-(12/6*3)+4)*3

postfix expression = 85*1263*/4+-3*

Q2: write a program for postfix Evalut -ion.

→

```
#include<stdio.h>
int stack[20];
int top = -1;
void push(int x)
{
    stack[++top] = x;
}
int pop()
{
    return stack[top--];
}
int main()
{
```

```
char exp[20];
char *e;
int n1, n2, n3, num;
printf("Enter the expression :: ");
    scanf("%s", exp);
      e = exp;

while (*e != '\0')
{
 if (isdigit(*e))
  {
    num = *e - 48;
    push(num);
  }
else
{
  n1 = pop();
  n2 = pop();
  switch (*e)
  {
  case '+';
  {
    n3 = n1 + n2;
      break;
  }
  case '-';
  {
    n3 = n1 + n2;
      break;
  }
```

```
case '*' ;
  {
    n3 = n1 * n2 ;
    break ;
  }
case '/' ;
  {
    n3 = n2 / n1 ;
    break ;
  }
}
push (n3);
}
e++ ;
}
printf (" \n The result of expression %s = %d \n\n"
                    exp, pop(1) ;
```

→ output:-

Enter expression : 12*34*+5-

the result of expression 12*34*+5- = 9

3] Q. WAP to simulate the working of a 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>
#define MAX 50
int queue_array [MAX];
 int rear = -1;
int front = -1;
display ()
 {
  int i;
  if (front == -1)
    printf ("queue is empty \n");
else
  {
    printf ("queue is : \n");
    for (i = front; i <= rear; i++)
    printf ("%d", queue_array [i]);
    printf ("\n");
  }
}
main ()
{
 int choice;
 while (1)
 {
   printf (" 1. insert \n");
   print f (" 2. delete \n");
   print f ("3. display \n");
```

```
printf ("4. exit \n");
printf ("Enter your choice :");
scanf ("%d", &choice);
switch (choice)
  {
   case 1 :
   insert ();
   break;
   case 2 :
   delete();
   break ;
   case 3 ;
   display ();
   break;
   case 4:
    exit();
    break ;
   default :
   printf ("invalid choice \n");
  }
}
}

insert ()
{
 int add_item ;
  if (rear == MAX-1)
    printf ("queue overflow \n");
 else
 {
  if (front == -1)
  front = 0 ;
```

```
printf ("insert the element in the queue :");
scanf ("%d", &add_item);
rear++;
queue_array [rear] = add_item;
}
}
delete()
{
if (front == -1 || front > rear)
{
printf ("queue underflow\n");
return;
}
else
{
printf ("deleted element is : %d\n",
queue_array [front]);
front += 1;
}
}
```

→ output :

1. insert
2. delete
3. display
4. exit

enter your choice : 1
insert the element in the queue : 15

1. insert
2. delete
3. display
4. exit

Date
Page

enter your choice : 2
deleted element is = 15
B :

2 8 12