

```

#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();
            }
    }
}

```

```

28     }
29     else
30     {
31         return(0);
32     }
33 }
34 void main()
35 {
36     char infix[50], postfix[50], ch, elem;
37     int i=0, k=0;
38     printf("enter the infix expression:");
39     scanf("%s", infix);
40     push('#');
41     while((ch=infix[i++]) != '\0')
42     {
43         if( ch =='(') push(ch);
44         else
45             if(isalnum(ch)) postfix[k++]=ch;
46         else
47             if(ch==')')
48             {
49                 while( stack[top] != '(')
50                     postfix[k++]=pop();
51                 elem=pop();
52             }
53         else
54         {
55             while(pr(stack[top])>=pr(ch))
56                 postfix[k++]=pop();
57             push(ch);
58         }
59     }
60     while(stack[top] != '#')
61         postfix[k++]=pop();
62     postfix[k]='\0';
63     printf("\n Postfix Expression = %s\n", postfix);
64 }
65

```

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.

```

1  #include<stdio.h>
2  int stack[20];
3  int top=-1;
4  void push(int x)
5  {
6      stack[++top]=x;
7  }
8  int pop()
9  {
10     return stack[top--];
11 }
12 int main()
13 {
14     char exp[20];
15     char *e;
16     int n1,n2,n3,num;
17     printf("Enter experssion ::");
18     scanf("%s",exp);
19     e=exp;
20
21     while(*e !='\0')
22     {
23         if(isdigit(*e))
24         {
25             num=*e -48;
26             push(num);
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;

```

```
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.
```

```

1  #include<stdio.h>
2  #define MAX 50
3  int queue_array[MAX];
4  int rear=-1;
5  int front=-1;
6  display()
7  {
8      int i;
9      if(front== -1)
10         printf("queue is empty\n");
11     else
12     {
13         printf("queue is : \n");
14         for(i=front; i<=rear; i++)
15             printf("%d", queue_array[i]);
16         printf("\n");
17     }
18 }
19 main()
20 {
21     int choice;
22     while(1)
23     {
24         printf("1.insert\n");
25         printf("2.delete\n");
26         printf("3.display\n");
27         printf("4.exit\n");
28         printf("enter your choice:");
29         scanf("%d", &choice);
30         switch(choice)
31         {
32             case 1:
33                 insert();
34                 break;
35             case 2:
36                 delete();
37                 break;
38             case 3:
39                 display();
40                 break;
41             case 4:
42                 exit(1);
43                 break;
44             default:
45                 printf("invalid choice\n");
46         }
47     }
48 }
49 insert()
50 {
51     int add_item;

```

```

8 }
9 insert()
10 {
11     int add_item;
12     if(rear==MAX-1)
13         printf("queue overflow\n");
14     else
15     {
16         if(front== -1)
17             front=0;
18         printf("insert the element in the queue:");
19         scanf("%d",&add_item);
20         rear+=1;
21         queue_array[rear]=add_item;
22     }
23 }
24 delete()
25 {
26     if(front== -1 || front>rear)
27     {
28         printf("queue underflow\n");
29         return;
30     }
31     else
32     {
33         printf("deleted element is : %d\n",queue_array[front]);
34         front+=1;
35     }
36 }
37
38
39

```



```
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 program to convert a given valid parenthesis-sized 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(u);
}
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);

```

```

                }
    }
}

```


Date / /
Page

```

}
while (stack[top] != '#')
    postfix[k++] = pop();
postfix[k] = '\0';
printf("In 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 Evaluation.

```

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

```

Date _____
Page _____

```

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

```

```

while (*e != '\0')
{

```

```

    if (is digit(*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",
exp, Pop());

```

→ output:-

Enter expression : 12*30*+5-
the result of expression 12*30*+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 overflow 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");
        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(0);
        break;
    default:
        printf("invalid choice\n");
}
```

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

of
wing
program
empty

Date: / /
Page:

```

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++;
    }
}
}

```

→ 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

enter your choice : 2

deleted element is = 15

is ?

28/12