PROGRAM

```
/*-----------------------------------------------------------------------------------------------------------*/
/*-----TITLE      : INFIX TO POSTFIX CONVERTION---------------------- */
/*-----AIM        : TO PERFORM INFIX TO POSTFIX CONVERTION----- */
/*-----AUTHOR  : ----------------------------------------------------------------------------------------*/
/*-----------------------------------------------------------------------------------------------------------*/


#include<iostream>
#include<stdlib.h>
#include<string.h>
#define size 100
using namespace std;
int top=-1;
class Infixpostfix
{

                char stack[size];
        public:
                void push(char ch);
                char pop();
                char topelement();
                int precedence(char ch);
                int braces(char *);
};

void Infixpostfix ::push(char ch)
{
        if(top!=size-1)
        {
                top++;
                stack[top]=ch;
        }
}


char Infixpostfix ::pop()
{
        char a;
        if(top!=-1)
        {
                a=stack[top];
                top--;
                return(a);
        }
```

```
        else
        {
                return('#');
        }
}
char Infixpostfix ::topelement()
{
        char ch;
        if(top!=-1)
        {
                ch=stack[top];

        }
        else
        {
                ch='#';
                return(ch);

        }
}


int Infixpostfix ::precedence(char ch)
{
        switch(ch)
        {
                case '^':return(5);
                        break;

                case '/':return(4);
                        break;

                case '*':return(4);
                        break;

                case '+':return(3);
                        break;

                case '-':return(3);
                        break;


                default :return(0);
                        break;

        }
```

```
}
int Infixpostfix ::braces(char *s)
{
        int lbr,rbr;
        lbr=rbr=0;
        for(int i=0;s[i];i++)
        {
                if(s[i]=='(')
                {
                        lbr++;
                }
                if(s[i]==')')
                {
                        rbr++;
                }
        }
        if(lbr==rbr)
        {
                return(0);
        }
        else if(lbr>rbr)
        {
                return(1);
        }
          else
            {
                return(-1);
            }

}

main()
{
        Infixpostfix obj;
        char postfix[size],infix[size],element,topelement,st[2];
        int j=0,chk=0,popped,preelement,ptopelement;
        strcpy(postfix," ");
        system("clear");
        cout<<"Enter the infix expression:\n";
        cin.getline(infix,size);
        chk=obj.braces(infix);
        if(chk!=0)
        {
                cout<<"Unbalanced number of brackets\n";
                exit(0);
        }
```

```
for(int i=0;infix[i]!='\0';i++)
 {
        if(infix[i]!='(' && infix[i]!=')' && infix[i]!='^' && infix[i]!='*' && infix[i]!='/'
            && infix[i]!='+' && infix[i]!='-' )
        {
                postfix[j++]=infix[i];
        }
        else if(infix[i]=='(')
        {
                element=infix[i];
                obj.push(element);
        }
           else if(infix[i]==')')
           {
                while((popped=obj.pop())!='(')
                {
                        postfix[j++]=popped;
                }
           }
                 else
                 {
                        element=infix[i];
                        preelement=obj.precedence(element);
                        topelement=obj.topelement();
                        ptopelement=obj.precedence(topelement);
                        if(preelement>ptopelement)
                        {
                                obj.push(element);
                        }
                        else
                        {
                                while(ptopelement>=preelement)
                                {
                                        if(topelement=='#')
                                        {
                                                break;
                                        }

                                        popped=obj.pop();
                                        postfix[j++]=popped;
                                        topelement=obj.topelement();

                                        ptopelement=obj.precedence(topelement);
                                }
                                obj.push(element);
                        }
```

```
                        }
        }
        while((popped=obj.pop())!='#')
        {
                postfix[j++]=popped;
        }

        postfix[j]='\0';
        cout<<"\n postfix:"<<postfix<<"\n";
        return(0);
}
```

OUTPUT

```
exam@minipc-Default-string:~$ g++ postfixevalu.cpp
exam@minipc-Default-string:~$ ./a.out
enter the postfix expression
636+5*9/-
result=1
```

PROGRAM

```
/*-------------------------------------------------------------------------------------------*/
/*-----TITLE       : POSTFIX EVALUATION------------------------------------ */
/*-----AIM         : TO PERFORM POSTFIX EVALUATION-----------------*/
/*-----AUTHOR   :-------------------------------------------------------------------------*/
/*-------------------------------------------------------------------------------------------*/


#include<iostream>
#include<stdlib.h>
#include<math.h>
#include<ctype.h>
using namespace std;
const int size=50;
class postfix
{
        private:
                int stack[size];
                int top,n;
                char *s;
        public:
                postfix();
                void setexpression(char * str);
                void push(int item);
                int pop();
                void calculate();
                void display();
};
postfix::postfix()
{
        top=-1;
}
void postfix::setexpression(char *str)
{
        s=str;
}
```

```
void postfix::push(int item)
 {
        if(top==size-1)
        {
                cout<<"stack is full\n";
                exit(0);
        }
        else
        {
                top++;
                stack[top]=item;
        }
 }
 int postfix::pop()
 {
        if(top==-1)
        {
                cout<<"stack is empty\n";
                exit(0);
        }
        else
        {
                int data=stack[top];
                top--;
                return (data);
        }
 }
 void postfix::calculate()
 {
        int n1,n2,n3;
        while(*s)
        {
                if(*s=='\t')
                {
                        s++;
                        continue;
                }
                if(isdigit(*s))
                {
                        n=*s-'0';
                        push(n);
                }
                else
                {
                        n1=pop();
                        n2=pop();
```

```
                        switch(*s)
                        {
                                case '+':n3=n2+n1;
                                        break;
                                case '-':n3=n2-n1;
                                        break;
                                case '/':n3=n2/n1;
                                        break;
                                case '*':n3=n2*n1;
                                        break;
                                case '%':n3=n2%n1;
                                        break;
                                case '^':n3=pow(n2,n1);
                                        break;
                                default: cout<<"unknown operator\n";
                                        exit(1);
                        }
                        push(n3);
                }
                s++;
        }
}
void postfix::display()
{
        n=pop();
        cout<<"result="<<n<<"\n";
}
int main()
{
        char expression[size];
        cout<<"enter the postfix expression\n";
        cin.getline(expression,size);
        postfix obj;
        obj.setexpression(expression);
        obj.calculate();
        obj.display();
        return (0);
}
```

OUTPUT

```
exam@minipc-Default-string:~$ g++ postfix_infix.cpp
exam@minipc-Default-string:~$ ./a.out
Enter the infix expression: a*b+(c/d)-e
 postfix:ab*cd/+e
```