Name: Shruti Mohan Baravkar
Exam seat no :
Roll no : SE265
Batch : B1

……

ASSIGNMENT NO : 11

Implement C++ program for expression conversion as infix to postfix and its evaluation using stack based on given conditions
i. Operands and operator, both must be single character.
ii. Input Postfix expression must be in a desired format.
iii. Only '+', '-', '*' and '/ ' operators are expected.

```
#include<iostream>

using namespace std;
class stack
{
 public:
  char stack_array[50];
  int top;
  stack()
  {
    top=-1;
  }
void push(char symbol)
{  if(full())
      cout<<"\nStack overflow:\n";
    else
    { top=top+1;
      stack_array[top]=symbol;
     }
}
char pop()
{   if(empty())
       return('#');         // Return value '#' indicates stack is empty
     else
       return(stack_array[top--]);
}
int empty()
{    if(top==-1)
       return(1);
     else
       return(0);
}
int full()
{    if(top==49)
       return(1);
     else
       return(0);
}
private:
  char infix[50];
```

```
  char postfix[50];
 public:
    void read()
    {
      cout<<"\nEnter an infix expression:";
      cin>>infix;
    }
int white_space(char symbol)
  {  if(symbol==' ' || symbol=='\t' || symbol=='\0')
         return 1;
      else
        return 0;
  }
void ConvertToPostfix()
 {    int prev,p;
      char entry;
      p=0;
      for(int i=0;infix[i]!='\0';i++)
      {
    if(!white_space(infix[i]))
    { switch(infix[i])
      {
        case '(': push(infix[i]);
                  break;
        case ')': while((entry=pop())!='(')
                  postfix[p++]=entry;
                  break;
        case '+':
        case '-':
        case '*':
        case '/':
        if(!empty())
          {  prev=prior(infix[i]);
             entry=pop();
         while(prev<=prior(entry))
         {  postfix[p++]=entry;
           if(!empty())
              entry=pop();
           else
              break;
         }
        if(prev>prior(entry))
           push(entry);
        }
          push(infix[i]);
          break;
          default:
          postfix[p++]=infix[i];
          break;
        }
      }
    }
      while(!empty())                //while stack is not empty
       postfix[p++]=pop();
    postfix[p]='\0';
    cout<<"\nThe postfix expression is: "<<postfix<<endl;
  }
int prior(char symbol)
```

```
{  switch(symbol)
    { case '/': return(4);          // Precedence of / is 4
      case '*': return(3);          // Precedence of * is 3
      case '+': return(2);          // Precedence of + is 2
      case '-': return(1);          // Precedence of - is 1
      case '(': return(0);          // Precedence of ( is 0
      default: return(-1);
    }
  }
};
int main()
{  char choice='y';
   stack expr;
   while(choice=='y')
  {expr.read();
   expr.ConvertToPostfix();
   cout<<"\n\nDo you want to continue ? (y/n): ";
   cin>>choice;
 }
 return 0;
}
```

OUTPUT:

```
Enter an infix expression:((a+b)*(c-(d/e)))
The postfix expression is: ab+cde/-*
Do you want to continue ? (y/n): n
```