**PRACTICAL NO:10**

**INPUT:**

#include<iostream>

#include<string.h>

#include<stack>

using namespace std;

int getweight(char ch){

switch(ch){

case '/':

case '\*':return 2;

case '+':

case '-':return 1;

default:return 0;

}

}

void infix2postfix(char infix[],char postfix[],int size){

stack<char> s;

int weight;

int i=0;

int k=0;

char ch;

while(i<size){

ch=infix[i];

if(ch=='(')

{

s.push(ch);

i++;

continue;

}

if(ch==')'){

while(!s.empty()&& s.top() !='('){

postfix[k++]=s.top();

s.pop();

}

if(!s.empty()){

s.pop();

}

i++;

continue;

}

weight=getweight(ch);

if(weight==0)

{

postfix[k++]=ch;

}

else

{

if(s.empty())

{

s.push(ch);

}

else

{

while(!s.empty() && s.top() !='(' && weight <= getweight(s.top()))

{

postfix[k++]=s.top();

s.pop();

}

s.push(ch);

}

}

i++;

}

while (!s.empty())

{

postfix[k++]=s.top();

s.pop();

}

postfix[k]=0;

}

int main()

{

char infix[100];

cout<<"\nEnter infix operation:";

cin>>infix;

int size=strlen(infix);

char postfix[size];

infix2postfix(infix,postfix,size);

cout<<"\nInfix Expression::"<<infix;

cout<<"\nPostfix Expression::"<<postfix;

cout<<endl;

return 0;

}

**OUTPUT:**

Enter infix operation:(A\*B+D)\*C-D

Infix Expression::(A\*B+D)\*C-D

Postfix Expression::AB\*D+C\*D-