**Experiment No: 3C.**

**Title :** Implementation of prefix to postfix expression using recursion.

**Problem Statement :** Convert the given prefix expression to postfix expression in C++ using recursion.

**Algorithm :**

* Read the Prefix expression in reverse order (from right to left)
* If the symbol is an operand, then push it onto the Stack
* If the symbol is an operator, then pop two operands from the Stack
* Create a string by concatenating the two operands and the operator after them.
* string = operand1 + operand2 + operator
* And push the resultant string back to Stack
* Repeat the above steps until end of Prefix expression.

**Code :**

#include<iostream>

#include<stack>

using namespace std;

string prefix;

string postfix;

int i;

int l;

stack<string> s;

bool isoperator(char ch)

{

switch(ch)

{

case '+' :

case '-' :

case '\*' :

case '/' :

return true;

}

return false;

}

void eval(int i)

{

if(i<0)

{

postfix = s.top();s.pop();

cout<<"Postfix expression :"<<postfix<<endl;

return;

}

if(isoperator(prefix[i]))

{

string p1 = s.top();s.pop();

string p2 = s.top();s.pop();

string temp = p1+p2+prefix[i];

s.push(temp);

}

else{

s.push(string(1,prefix[i]));

}

i = i - 1;

eval(i);

}

int main()

{

cout<<"enter the prefix expression :";

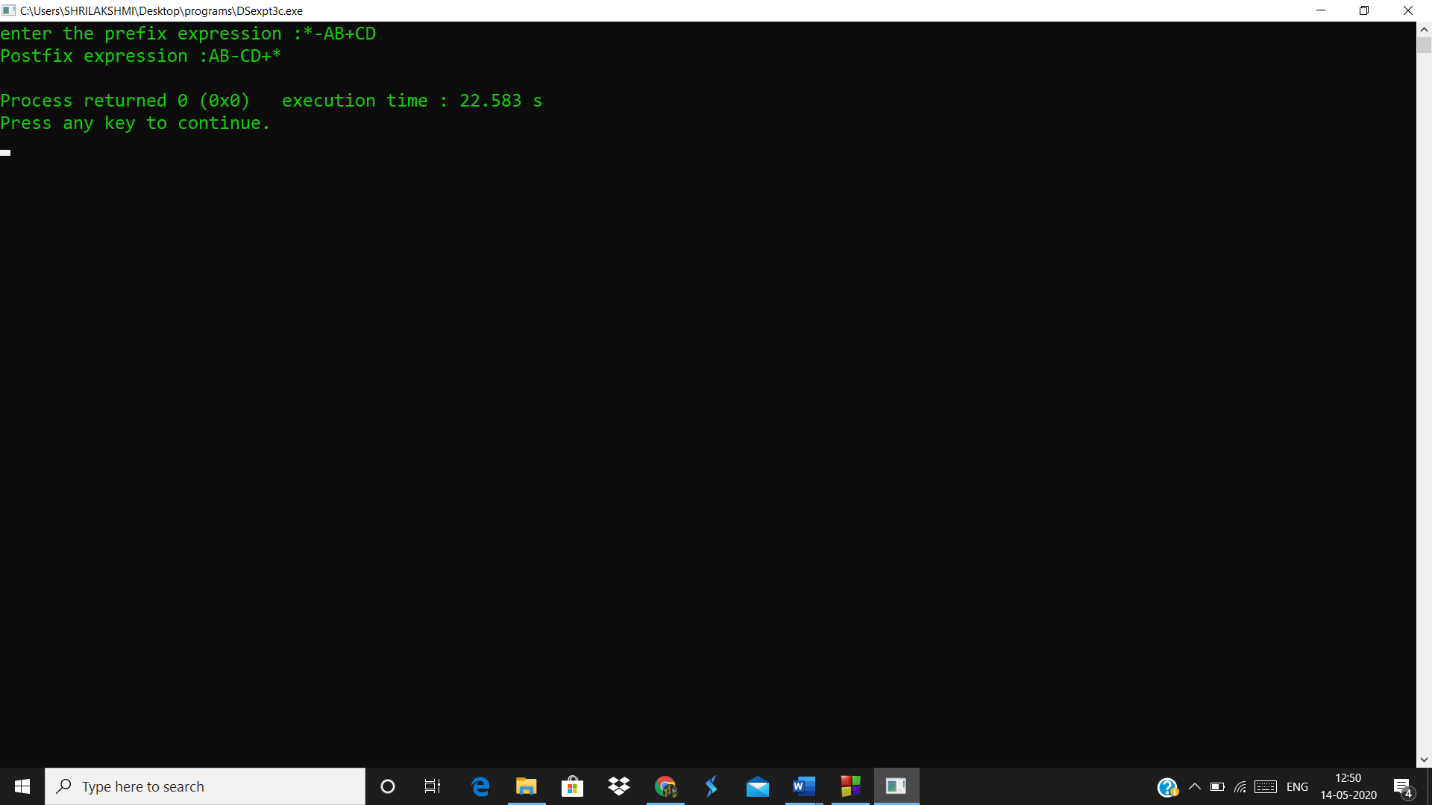
cin>>prefix;

l = prefix.size();

eval(l-1);

}

**Output :**

****

**Analysis :**

* The program checks if character is an operator or not but if the expression has numbers or the variables both are treated similarly rather than returning the answer of expression with numbers.
* This code is feasible in C++ since we declare the stack using header and concatenation of strings is simple in C++ as compared to C.
* Since we use recursion the code is a bit inefficient as compared to loop.