**Experiment: 02**

**Objective:** Write a program to implement  removal of left recursion

**Theory:** If the leftmost symbol on the right side is equal to the nonterminal on the left side, the generation is left-recursive. For instance, exp + term. A Left-Recursive Grammar is a grammar that contains a production with left recursion. Right-recursion occurs when the rightmost symbol on the right side equals the left side.

Consider an example: E->E+T/T,

The above example will go in an infinite loop because the function E keeps calling itself which causes a problem for a parser to go in an infinite loop which is a never-ending process.

so to avoid this infinite loop problem we do Elimination of left recursion.

Rules to follow to eliminate left recursion

       A-->bA'

    A'-->eps/aA'     //eps stands for epsilon

Therefore solution for the above left recursion problem,

E --> TE'

           E'-->eps/+TE'

**Source Code:**

/\*

E->iE'

E'->+iE'/e;

\*/

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

void X(int& l,string check){

if(check[l]!='+')

return;

l++;

if(check[l]!='i')

return;

l++;

X(l,check);

}

void E(int& l, string check){

if(check[l]!='i')

return;

l++;

X(l,check);

}

int main()

{

string check="i+i$";

int l=0;

E(l,check);

if(check[l]=='$')

cout<<"It is accepted by this grammar";

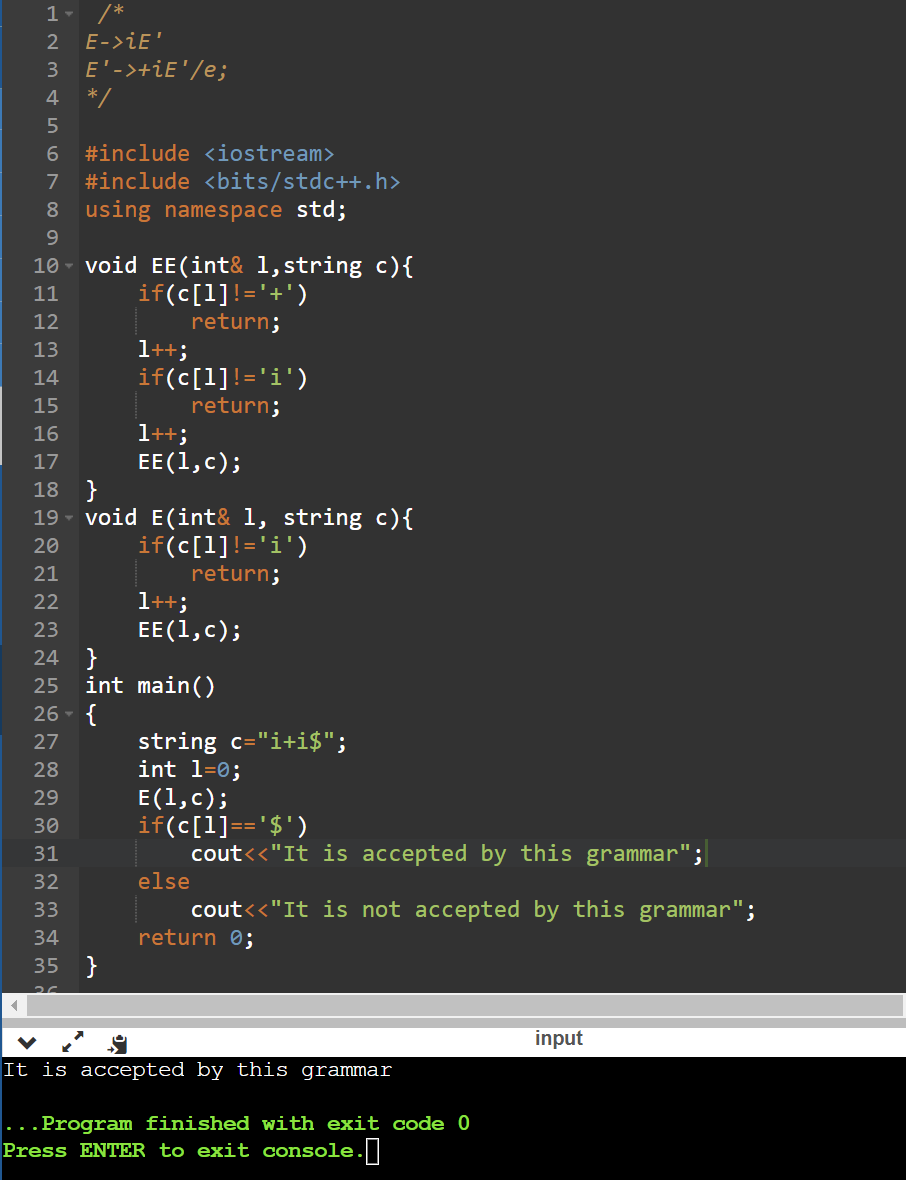
else

cout<<"It is not accepted by this grammar";

return 0;

}

**OUTPUT:**

****