SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

FACULTY OF ENGINEERING & TECHNOLOGY

(Formerly SRM University, Under section 3 of UGC Act, 1956)

S.R.M. NAGAR, KATTANKULATHUR –603 203, KANCHEEPURAM DISTRICT

SCHOOL OF COMPUTING

DEPARTMENT OF NETWORKING AND COMMUNICATIONS

LAB RECORD

Course code: 18CSC304J
Course name: Compiler Design

Name: Praveen Kumar K

Registration number: RA1911030010069
Section: O2
Branch: CSE- Cyber Security

INDEX

Name: Praveen Kumar K
Reg no: RA1911030010069
Subject: Compiler Design

Exp1: Lexical Analyser

Code:

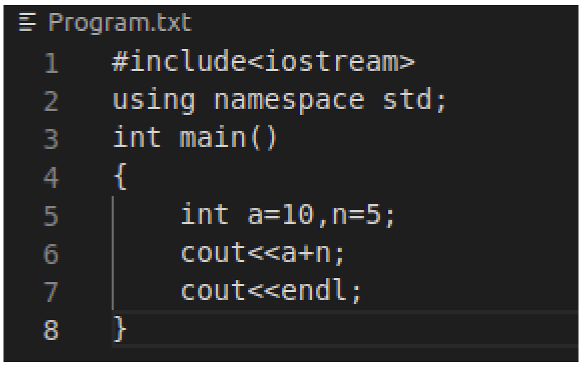
```
#include<bits/stdc++.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>

using namespace std;

int isKeyword(char buffer[]){
    char keywords[32][10] =
    {"auto","break","case","char","const","continue","default",

"do","double","else","enum","extern","float","for","goto",

"if","int","long","register","return","short","signed",

"sizeof","static","struct","switch","typedef","union",
                            "unsigned","void","volatile","while"};
    int i, flag = 0;

    for(i = 0; i < 32; ++i){
        if(strcmp(keywords[i], buffer) == 0){
            flag = 1;
            break;
        }
    }

    return flag;
}

int main(){
    char ch, buffer[15],b[30], logical_op[] = "><",math_op[]="+-
*/=",numer[]=".0123456789",other[]=",;\(){}[]":";
    ifstream fin("Program.txt");
    int mark[1000]={0};
    int i,j=0,kc=0,ic=0,lc=0,mc=0,nc=0,oc=0,aaa=0;
    vector < string > k;
    vector<char >id;
    vector<char>lo;
    vector<char>ma;
    vector<string>nu;
     vector<char>ot;
    if(!fin.is_open()){
        cout<<"error while opening the file\n";
        exit(0);
    }

    while(!fin.eof()){
           ch = fin.get();
          for(i = 0; i < 12; ++i){
               if(ch == other[i]){
                    int aa=ch;
                if(mark[aa]!=1){
                    ot.push_back(ch);
                    mark[aa]=1;
                ++oc;
             }
            }
        }

        for(i = 0; i < 5; ++i){
            if(ch == math_op[i]){
                int aa=ch;
             if(mark[aa]!=1){
                ma.push_back(ch);
                mark[aa]=1;
                ++mc;
             }
            }
        }
        for(i = 0; i < 2; ++i){
            if(ch == logical_op[i]){
                int aa=ch;
             if(mark[aa]!=1){
                lo.push_back(ch);
                mark[aa]=1;
                ++lc;
             }
            }

        }
        if(ch=='0' || ch=='1' || ch=='2' || ch=='3' || ch=='4' || ch=='5' ||
ch=='6' || ch=='7' || ch=='8' || ch=='9' || ch=='.' ||ch == ' ' || ch == '\n'
|| ch == ';'){

            if(ch=='0' || ch=='1' || ch=='2' || ch=='3' || ch=='4' || ch=='5' ||
ch=='6' || ch=='7' || ch=='8' || ch=='9' || ch=='.')b[aaa++]=ch;
```

```
            if((ch == ' ' || ch == '\n' || ch == ';') && (aaa != 0)){
                b[aaa] = '\0';
                aaa = 0;
                char arr[30];
                 strcpy(arr,b);
                     nu.push_back(arr);
                 ++nc;

              }
          }



           if(isalnum(ch)){
               buffer[j++] = ch;
           }
           else if((ch == ' ' || ch == '\n') && (j != 0)){
                buffer[j] = '\0';
                j = 0;

                if(isKeyword(buffer) == 1){

                    k.push_back(buffer);
                    ++kc;
                }
                else{



                 if(buffer[0]>=97 && buffer[0]<=122) {
                     if(mark[buffer[0]-'a']!=1){
                     id.push_back(buffer[0]);
                     ++ic;
```

```
                          mark[buffer[0]-'a']=1;
                         }

                      }

                     }

               }

      }

      fin.close();
      printf("Keywords: ");
       for(int f=0;f<kc;++f){
             if(f==kc-1){
                 cout<<k[f]<<"\n";
             }
             else {
                 cout<<k[f]<<", ";
             }
      }
      printf("\nIdentifiers: ");
       for(int f=0;f<ic;++f){
          if(f==ic-1){
                 cout<<id[f]<<"\n";
             }
             else {
                 cout<<id[f]<<", ";
             }
      }
      printf("\nMath Operators: ");
      for(int f=0;f<mc;++f){
             if(f==mc-1){
                 cout<<ma[f]<<"\n";
             }
             else {
                 cout<<ma[f]<<", ";
             }
      }
      printf("\nLogical Operators: ");
      for(int f=0;f<lc;++f){
             if(f==lc-1){
                 cout<<lo[f]<<"\n";
             }
             else {
                 cout<<lo[f]<<", ";
             }

      }
      printf("\nNumerical Values: ");
      for(int f=0;f<nc;++f){
             if(f==nc-1){
                 cout<<nu[f]<<"\n";
             }
             else {
                 cout<<nu[f]<<", ";
             }

      }
      printf("\nOthers: ");
      for(int f=0;f<oc;++f){
             if(f==oc-1){
                 cout<<ot[f]<<"\n";
             }
```

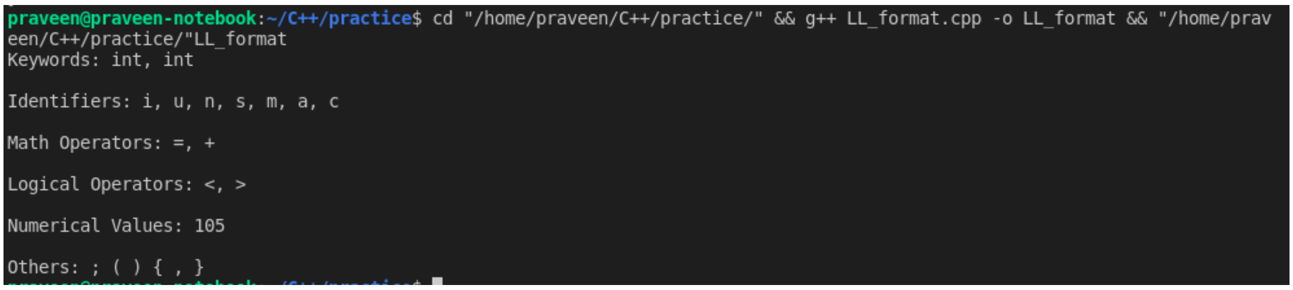
```
            else {
                cout<<ot[f]<<" ";
            }

        }

    return 0;
}
```

Output:



Program.txt

```
#include<iostream>
using namespace std;
int main()
{
    int a=10,n=5;
    cout<<a+n;
    cout<<endl;
}
```



```
praveen@praveen-notebook:~/C++/practice$ cd "/home/praveen/C++/practice/" && g++ LL_format.cpp -o LL_format && "/home/praveen/C++/practice/"LL_format
Keywords: int, int

Identifiers: i, u, n, s, m, a, c

Math Operators: =, +

Logical Operators: <, >

Numerical Values: 105

Others: ; ( ) { , }
```

Results:

Hence the all the keywords, identifiers and others have been identified.

Name: Praveen Kumar K
Reg no: RA1911030010069
Subject: Compiler Design

EXP2: Regular Expression to NFA

AIM:

Conersion of Regular expression to NFA

Algorithm:

1. Start
2. Get the input from the user
3. Initialize separate variables and functions for Postfix , Display and NFA
4. Create separate methods for different operators like +,*, .
5. By using Switch case Initialize different cases for the input
6. For ' . ' operator Initialize a separate method by using various stack functions do the
same for the other operators like ' * ' and ' + '.
7. Regular expression is in the form like a.b (or) a+b
8. Display the output
9. Stop

Code:

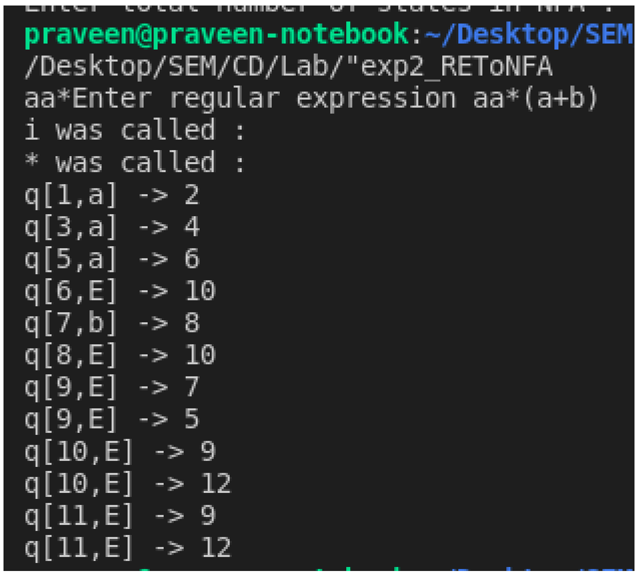
```
#include<bits/stdc++.h>

using namespace std;
```

```
int prio(char c) {
        if(c == '*') return 3;
        if(c == '.') return 2;
        if(c == '+') return 1;
        return 0;
}

string infix_to_postfix(string s) {
        stack<char> st;
        st.push('(');
        s += ')';
        int n = s.size();
        string ret;
        for(int i=0; i<n; i++) {
                if(prio(s[i])) {
                        while(!st.empty()) {
                                if(prio(st.top()) <= prio(s[i])) {
                                        st.push(s[i]);
                                        break;
                                }
                                ret += st.top();
                                st.pop();
                        }
                } else if(s[i] == '(') {
                        st.push(s[i]);
                } else if(s[i] == ')') {
                        while(!st.empty()) {
                                if(st.top() == '(') {
                                        st.pop();
                                        break;
                                }
                        ret += st.top();
                        st.pop();
                  }
            } else if(isalpha(s[i])) {
                  ret += s[i];
            }
      }
      return ret;
}
int nodes = 0;
int ALPHABET_SZ = 3;
vector<vector<vector<int>>>
transitions(30,vector<vector<int>>(ALPHABET_SZ+1,vector<int
>(0)));

pair<int,int> handlePlus(int ss1,int es1,int ss2,int es2)
{
   cout<<"i was called : "<<endl;
   transitions[nodes+1][ALPHABET_SZ].push_back(ss1);
   transitions[nodes+1][ALPHABET_SZ].push_back(ss2);
   transitions[es1][ALPHABET_SZ].push_back(nodes+2);
   transitions[es2][ALPHABET_SZ].push_back(nodes+2);
   nodes = nodes+2;
   return make_pair(nodes-1,nodes);
}

pair<int,int> handleStar(int ss,int es)
{
   cout<<"* was called : "<<endl;
   transitions[es][ALPHABET_SZ].push_back(ss);
   transitions[nodes+1][ALPHABET_SZ].push_back(ss);
   transitions[nodes+1][ALPHABET_SZ].push_back(nodes+2);
```

```
    transitions[es][ALPHABET_SZ].push_back(nodes+2);
    nodes = nodes+2;
    return make_pair(nodes-1,nodes);
}

pair<int,int> handleDot(int ss2,int es2,int ss1,int es1)
{
    // cout<<"ss1 : "<<ss1;
    // cout<<"es1 : "<<es1;
    // cout<<"ss2 : "<<ss2;
    // cout<<"es2 : "<<es2;

    // cout<<". was called :"<<endl;
    transitions[es1][ALPHABET_SZ].push_back(ss2);
    return make_pair(ss1,es2);
}

pair<int,int> handleSingle(int alphabet)
{
    transitions[nodes+1][alphabet-'a'].push_back(nodes+2);
    nodes = nodes+2;
    return make_pair(nodes-1,nodes);
}
void postfix_to_e_nfa(string s)
{
    stack<pair<int,int>> st;
    for(int i=0;i<s.length();i++)
    {
        // cout<<"NODES : "<<nodes<<endl;
        if(s[i] == '*')
        {
            pair<int,int> r = st.top();
            st.pop();
            st.push(handleStar(r.first,r.second));
        }
        else if(s[i]=='+')
        {
            pair<int,int> r1 = st.top();
            st.pop();
            pair<int,int> r2 = st.top();
            st.pop();
            st.push(handlePlus(r1.first,r1.second,r2.first,r2.second));
        }
        else if(s[i]=='.')
        {
            cout<<"NODES BEFORE CALL : "<<nodes<<endl;
            pair<int,int> r1 = st.top();
            st.pop();
            pair<int,int> r2 = st.top();
            st.pop();
            st.push(handleDot(r1.first,r1.second,r2.first,r2.second));
            cout<<" NODES AFTER CALL : "<<nodes<<endl;
        }
        else
        {
            st.push(handleSingle(s[i]));
        }
    }
}

int main()
{
    cout << "Enter regular expression : ";
        string regex; cin >> regex;
```

```
      regex = infix_to_postfix(regex);
   // cout<<"REGULAR  EXPRESSION : "<<regex<<endl;
   postfix_to_e_nfa(regex);
   for(int i=0;i<=nodes;i++)
   {
      for(int j=0;j<=ALPHABET_SZ;j++)
      {
         char c = 'a';
         for(int k=0;k<transitions[i][j].size();k++)

cout<<"q["<<i<<","<<(j==ALPHABET_SZ?'E':char(c+j))<<"] ->
"<<transitions[i][j][k]<<"  "<<endl;
      }
   }
   return 0;
}
```

Output:



```
praveen@praveen-notebook:~/Desktop/SEM
/Desktop/SEM/CD/Lab/"exp2_REToNFA
aa*Enter regular expression aa*(a+b)
i was called :
* was called :
q[1,a] -> 2
q[3,a] -> 4
q[5,a] -> 6
q[6,E] -> 10
q[7,b] -> 8
q[8,E] -> 10
q[9,E] -> 7
q[9,E] -> 5
q[10,E] -> 9
q[10,E] -> 12
q[11,E] -> 9
q[11,E] -> 12
```

Name: Praveen Kumar K
Reg no: RA1911030010069
Subject: Compiler Design

EXP3: NFA to DFA

AIM:

To convert the given NFA to DFA

Algorithm:

```
1. Start
2. Get the input from the user
3. Set the only state in SDFA to “unmarked”.
4. while SDFA contains an unmarked state do:
a. Let T be that unmarked state
b. for each a in % do S = e-Closure(MoveNFA(T,a))
c. if S is not in SDFA already then, add S to SDFA (as an
“unmarked” state)
d. Set MoveDFA(T,a) to S
5. For each S in SDFA if any s & S is a final state in the NFA then,
mark S an a final state in
the DFA
6. Print the result.
7. Stop the program
```

Code:

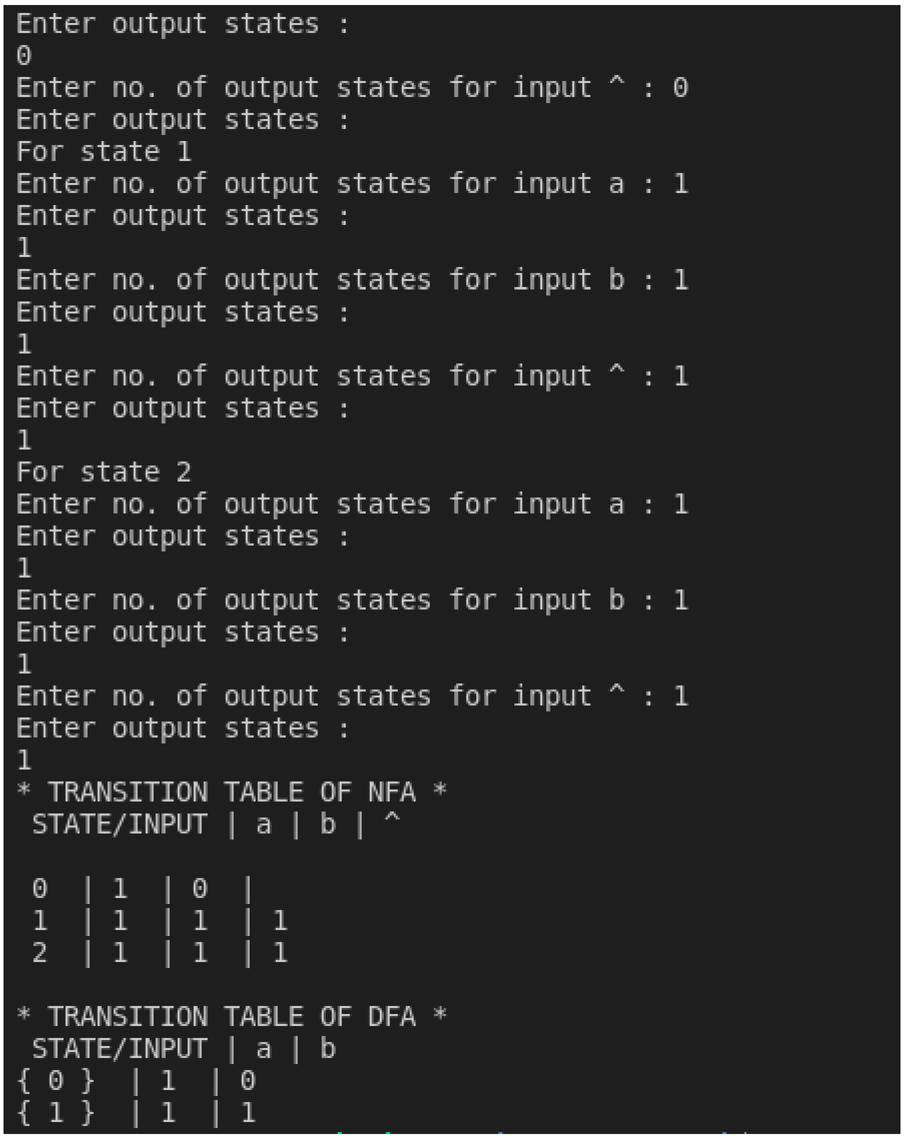
```
#include <iostream>
#include <bits/stdc++.h>

using namespace std;
```

```
void print(vector<vector<vector<int>>> table)
{
    cout << " STATE/INPUT |";
    char a = 'a';
    for (int i = 0; i < table[0].size() - 1; i++)
    {
        cout << " " << a++ << " |";
    }
    cout << " ^ " << endl
         << endl;
    for (int i = 0; i < table.size(); i++)
    {
        cout << " " << i << " ";
        for (int j = 0; j < table[i].size(); j++)
        {
            cout << " | ";
            for (int k = 0; k < table[i][j].size(); k++)
            {
                cout << table[i][j][k] << " ";
            }
        }
        cout << endl;
    }
}
void printdfa(vector<vector<int>> states,
vector<vector<vector<int>>> dfa)
{
    cout << " STATE/INPUT ";
    char a = 'a';
    for (int i = 0; i < dfa[0].size(); i++)
    {
        cout << "| " << a++ << " ";
    }
    cout << endl;
    for (int i = 0; i < states.size(); i++)
    {
        cout << "{ ";
        for (int h = 0; h < states[i].size(); h++)
            cout << states[i][h] << " ";
        if (states[i].empty())
        {
            cout << "^ ";
        }
        cout << "} ";
        for (int j = 0; j < dfa[i].size(); j++)
        {
            cout << " | ";
            for (int k = 0; k < dfa[i][j].size(); k++)
            {
                cout << dfa[i][j][k] << " ";
            }
            if (dfa[i][j].empty())
            {
                cout << "^ ";
            }
        }
        cout << endl;
    }
}
vector<int> closure(int s, vector<vector<vector<int>>> v)
{
    vector<int> t;
    queue<int> q;
    t.push_back(s);
    int a = v[s][v[s].size() - 1].size();
    for (int i = 0; i < a; i++)
    {
        t.push_back(v[s][v[s].size() - 1][i]);
        // cout<<"t[i]"<<t[i]<<endl;
        q.push(t[i]);
    }
    while (!q.empty())
    {
        int f = q.front();
        q.pop();
        if (!v[f][v[f].size() - 1].empty())
        {
            int u = v[f][v[f].size() - 1].size();
            for (int i = 0; i < u; i++)
            {
                int y = v[f][v[f].size() - 1][i];
                if (find(t.begin(), t.end(), y) == t.end())
                {
                    // cout<<"y"<<y<<endl;
                    t.push_back(y);
                    q.push(y);
                }
            }
        }
    }
    return t;
}
int main()
{
    int n, alpha;
    cout << "* NFA to DFA***" << endl
```

```
         << endl;
    cout << "Enter total number of states in NFA : ";
    cin >> n;
    cout << "Enter number of elements in alphabet(no of input
symbols) : ";
    cin >> alpha;
    vector<vector<vector<int>>> table;
    for (int i = 0; i < n; i++)
    {
        cout << "For state " << i << endl;
        vector<vector<int>> v;
        char a = 'a';
        int y, yn;
        for (int j = 0; j < alpha; j++)
        {
            vector<int> t;
            cout << "Enter no. of output states for input " << a++ <<
" : ";
            cin >> yn;
            cout << "Enter output states :" << endl;
            for (int k = 0; k < yn; k++)
            {
                cin >> y;
                t.push_back(y);
            }
            v.push_back(t);
        }
        vector<int> t;
        cout << "Enter no. of output states for input ^ : ";
        cin >> yn;
        cout << "Enter output states :" << endl;
        for (int k = 0; k < yn; k++)
        {
            cin >> y;
            t.push_back(y);
        }
        v.push_back(t);
        table.push_back(v);
    }
    cout << "* TRANSITION TABLE OF NFA *" << endl;
    print(table);
    cout << endl
         << "* TRANSITION TABLE OF DFA *" << endl;
    vector<vector<vector<int>>> dfa;
    vector<vector<int>> states;
    states.push_back(closure(0, table));
    queue<vector<int>> q;
    q.push(states[0]);
    while (!q.empty())
    {
        vector<int> f = q.front();
        q.pop();
        vector<vector<int>> v;
        for (int i = 0; i < alpha; i++)
        {
            vector<int> t;
            set<int> s;
            for (int j = 0; j < f.size(); j++)
            {
                for (int k = 0; k < table[f[j]][i].size(); k++)
                {
                    vector<int> cl = closure(table[f[j]][i][k], table);
                    for (int h = 0; h < cl.size(); h++)
                    {
```

```
                    if (s.find(cl[h]) == s.end())
                        s.insert(cl[h]);
                }
            }
        }
        for (set<int>::iterator u = s.begin(); u != s.end(); u++)
            t.push_back(*u);
        v.push_back(t);
        if (find(states.begin(), states.end(), t) == states.end())
        {
            states.push_back(t);
            q.push(t);
        }
    }
    dfa.push_back(v);
  }
  printdfa(states, dfa);
}
```

Output:



```
Enter output states :
0
Enter no. of output states for input ^ : 0
Enter output states :
For state 1
Enter no. of output states for input a : 1
Enter output states :
1
Enter no. of output states for input b : 1
Enter output states :
1
Enter no. of output states for input ^ : 1
Enter output states :
1
For state 2
Enter no. of output states for input a : 1
Enter output states :
1
Enter no. of output states for input b : 1
Enter output states :
1
Enter no. of output states for input ^ : 1
Enter output states :
1
* TRANSITION TABLE OF NFA *
 STATE/INPUT | a | b | ^

 0  | 1  | 0  |
 1  | 1  | 1  | 1
 2  | 1  | 1  | 1

* TRANSITION TABLE OF DFA *
 STATE/INPUT | a | b
{ 0 }  | 1  | 0
{ 1 }  | 1  | 1
```

Results:

Hence we have implemented the lexical analyser, Regular Expression to NFA and NFA to DFA

Name: Praveen Kumar K
Reg no: RA1911030010069
Subject: Compiler Design
Date: 17.02.2022

EXP: 4

AIM:

A program to eliminate Left Recursion using c++.

ALGORITHM:

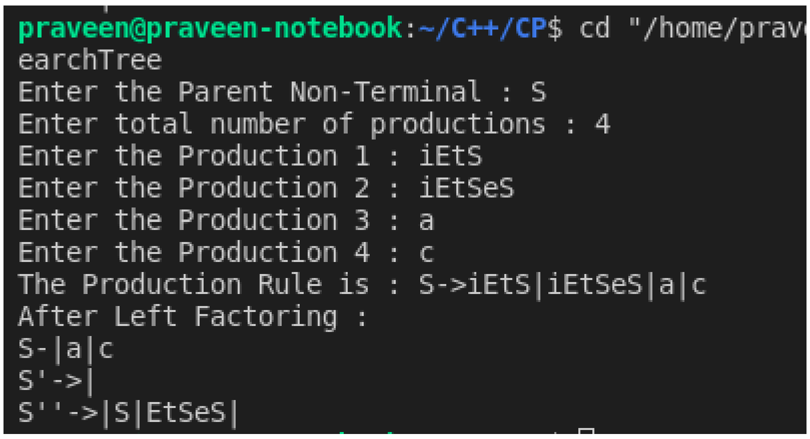
1. Start the program.
2. Initialize the arrays for taking input from the user.
3. Prompt the user to input the no. of non-terminals having left recursion and no. of
productions for these non-terminals.
4. Prompt the user to input the production for non-terminals.
5. Eliminate left recursion using the following rules:-
A->Aα1| Aα2 | . . . . . |Aαm
A->β1| β2| . . . . .| βn
Then replace it by
A-> βi A’ i=1,2,3,…..m
A’-> αj A’ j=1,2,3,…..n
A’-> Ɛ
6. After eliminating the left recursion by applying these rules, display the productions
without left recursion.

7. Stop.

CODE:

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;

int main()
{
    int n;
    cout<<"\nEnter number of non terminals: ";
    cin>>n;
    cout<<"\nEnter non terminals one by one: ";
    int i;
    vector<string> nonter(n);
    vector<int> leftrecr(n,0);
    for(i=0;i<n;++i) {
            cout<<"\nNon terminal "<<i+1<<" : ";
        cin>>nonter[i];
    }
    vector<vector<string> > prod;
    cout<<"\nEnter 'esp' for null";
    for(i=0;i<n;++i) {
        cout<<"\nNumber of "<<nonter[i]<<" productions: ";
        int k;
        cin>>k;
        int j;
        cout<<"\nOne by one enter all "<<nonter[i]<<" productions";
        vector<string> temp(k);
        for(j=0;j<k;++j) {
            cout<<"\nRHS of production "<<j+1<<": ";
            string abc;
            cin>>abc;
            temp[j]=abc;

if(nonter[i].length()<=abc.length()&&nonter[i].compare(abc.subst
r(0,nonter[i].length()))==0)
                leftrecr[i]=1;
        }
        prod.push_back(temp);
    }
    for(i=0;i<n;++i) {
        cout<<leftrecr[i];
    }
    for(i=0;i<n;++i) {
        if(leftrecr[i]==0)
            continue;
        int j;
        nonter.push_back(nonter[i]+"'");
        vector<string> temp;
        for(j=0;j<prod[i].size();++j) {
            if(nonter[i].length()<=prod[i]
[j].length()&&nonter[i].compare(prod[i]
[j].substr(0,nonter[i].length()))==0) {
                string abc=prod[i][j].substr(nonter[i].length(),prod[i]
[j].length()-nonter[i].length())+nonter[i]+"'";
                temp.push_back(abc);
                prod[i].erase(prod[i].begin()+j);
                --j;
            }
            else {
                prod[i][j]+=nonter[i]+"'";
            }
        }
        temp.push_back("esp");
        prod.push_back(temp);
    }
    cout<<"\n\n";
    cout<<"\nNew set of non-terminals: ";
    for(i=0;i<nonter.size();++i)
        cout<<nonter[i]<<" ";
    cout<<"\n\nNew set of productions: ";
    for(i=0;i<nonter.size();++i) {
        int j;
        for(j=0;j<prod[i].size();++j) {
            cout<<"\n"<<nonter[i]<<" -> "<<prod[i][j];
        }
    }
    return 0;
}
```

OUTPUT:



```
praveen@praveen-notebook:~/C++/CP$ cd "/home/prav
earchTree
Enter the Parent Non-Terminal : S
Enter total number of productions : 4
Enter the Production 1 : iEtS
Enter the Production 2 : iEtSeS
Enter the Production 3 : a
Enter the Production 4 : c
The Production Rule is : S->iEtS|iEtSeS|a|c
After Left Factoring :
S-|a|c
S'->|
S''->|S|EtSeS|
```

RESULT:

Left Recursion has been implemented successfully using c++

Exp 4b: Implementation of Left Factoring Elimination

AIM:

To implement a c++ code for left factoting elimination

ALGORITHM:

1. Start
2. Ask the user to enter the set of productions
3. Check for common symbols in the given set of productions by comparing with:
A->aB1|aB2
4. If found, replace the particular productions with:
A->aA’

A’->B1 | B2|ε
5. Display the output
6. Exit

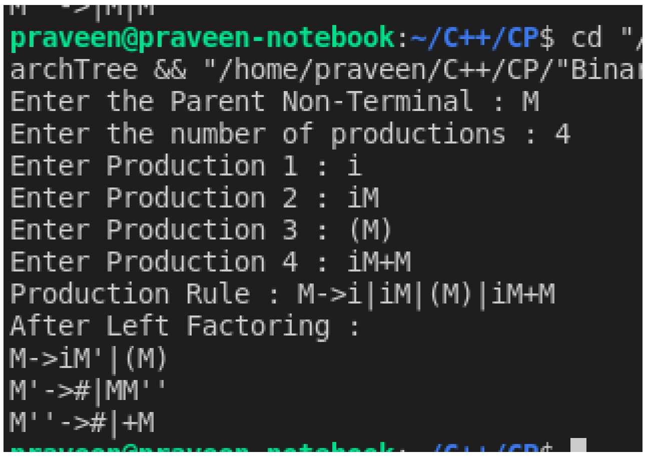
CODE:

```
#include<iostream>
#include<string>
using namespace std;
int main()
{  string ip,op1,op2,temp;
   int sizes[10] = {};
   char c;
   int n,j,l;
   cout<<"Enter the Parent Non-Terminal : ";
   cin>>c;
   ip.push_back(c);
   op1 += ip + "\'->";
   op2 += ip + "\'\'->";;
   ip += "->";
   cout<<"Enter the number of productions : ";
   cin>>n;
   for(int i=0;i<n;i++)
   {
      cout<<"Enter Production "<<i+1<<" : ";
      cin>>temp;
      sizes[i] = temp.size();
      ip+=temp;
      if(i!=n-1)
         ip += "|";
   }
```

```
cout<<"Production Rule : "<<ip<<endl;
char x = ip[3];
for(int i=0,k=3;i<n;i++)
{
   if(x == ip[k])
   {
         if(ip[k+1] == '|')
         {
            op1 += "#";
            ip.insert(k+1,1,ip[0]);
            ip.insert(k+2,1,'\'');
            k+=4;
         }
         else
         {
            op1 += "|" + ip.substr(k+1,sizes[i]-1);
            ip.erase(k-1,sizes[i]+1);
         }
   }
   else
   {
      while(ip[k++]!='|');
   }
}
char y = op1[6];
for(int i=0,k=6;i<n-1;i++)
{
   if(y == op1[k])
   {
         if(op1[k+1] == '|')
         {
            op2 += "#";
```

```
                op1.insert(k+1,1,op1[0]);
                op1.insert(k+2,2,'\");
                k+=5;
            }
            else
            {
                temp.clear();
                for(int s=k+1;s<op1.length();s++)
                    temp.push_back(op1[s]);
                op2 += "|" + temp;
                op1.erase(k-1,temp.length()+2);
            } }}
    op2.erase(op2.size()-1);
    cout<<"After Left Factoring : "<<endl;
    cout<<ip<<endl;
    cout<<op1<<endl;
    cout<<op2<<endl;
    return 0;
}
```

OUTPUT:



```
praveen@praveen-notebook:~/C++/CP$ cd "/
archTree && "/home/praveen/C++/CP/"Binar
Enter the Parent Non-Terminal : M
Enter the number of productions : 4
Enter Production 1 : i
Enter Production 2 : iM
Enter Production 3 : (M)
Enter Production 4 : iM+M
Production Rule : M->i|iM|(M)|iM+M
After Left Factoring :
M->iM'|(M)
M'->#|MM''
M''->#|+M
```

RESULT:

Hence the implementation of left factoring elimination has been done successfully using c++ :)

Name: Praveen Kumar K
Reg no: RA1911030010069
Subject: Compiler Design
Date: 24.02.2022

TITLE: FIRST AND FOLLOW

AIM:

A program to implement first follow in C++

ALGORITHM:

FIRST(A):

First(a) = set of terminals that start a string of terminals derived from a.

Apply following rules until no terminal or ε can be added

1. If t ∈ T, then First(t) = { t }.

2. If X ∈ N and X → ε exists (nullable), then add ε to First(X).

3. If X ∈ N and X → Y1Y2Y3... Ym, where Y1, Y2, Y3, ... Ym are non-terminals, then: for each, I from 1 to m if Y1 ... Yi-1 are all nullable (or if i = 1) First(X) = First(X) ∪ First(Yi)

FOLLOW(A):

Apply the following rules until no terminal or e can be added

1. $ ∈ Follow(S), where S is the start symbol.

2. Look at the occurrence of a non-terminal on the right-hand side of a production which is followed by something If A → a B b, then First(b) - {e} ⊆ Follow(B)

3. Look at N on the RHS that is not followed by anything, if (A → a B) or (A → a B b and ε ▯ First(b)), then Follow(A) ⊆ Follow(B)

CODE:

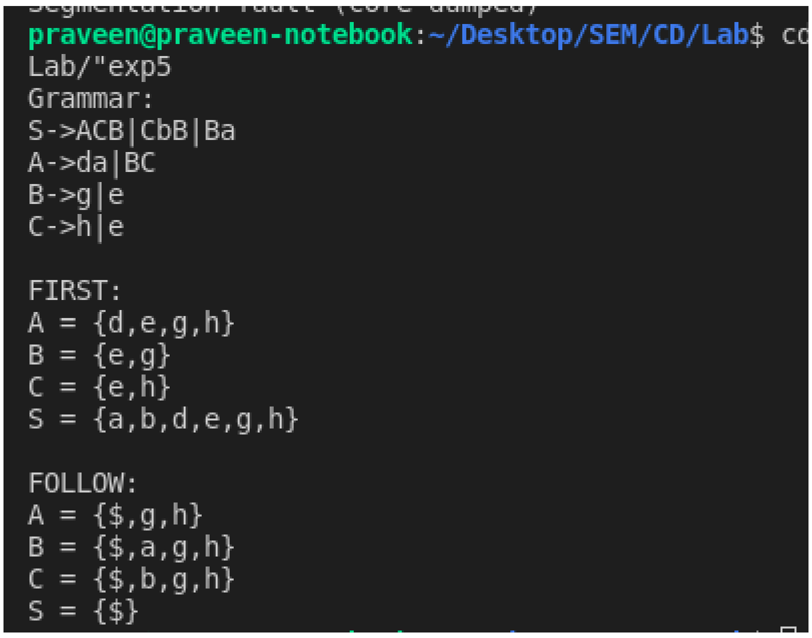
```
#include<bits/stdc++.h>
using namespace std;

set<char> ss;
bool dfs(char i, char org, char last, map<char,vector<vector<char>>> &mp){
    bool rtake = false;
    for(auto r : mp[i]){
        bool take = true;
        for(auto s : r){
            if(s == i) break;
            if(!take) break;
            if(!(s>='A'&&s<='Z')&&s!='e'){
                ss.insert(s);
                break;
            }
            else if(s == 'e'){
                if(org == i||i == last)
                ss.insert(s);
                rtake = true;
                break;
            }
            else{
                take = dfs(s,org,r[r.size()-1],mp);
                rtake |= take;
            }
        }
    }
    return rtake;
}

int main(){
    int i,j;
    ifstream fin("exp5_inputfirstfollow.txt");
    string num;
    vector<int> fs;
    vector<vector<int>> a;
    map<char,vector<vector<char>>> mp;
    char start;
    bool flag = 0;
    cout<<"Grammar: "<<'\n';
    while(getline(fin,num)){
        if(flag == 0) start = num[0],flag = 1;
        cout<<num<<'\n';
        vector<char> temp;
        char s = num[0];
        for(i=3;i<num.size();i++){
            if(num[i] == '|'){
                mp[s].push_back(temp);
                temp.clear();
            }
            else temp.push_back(num[i]);
        }
        mp[s].push_back(temp);
    }
    map<char,set<char>> fmp;
    for(auto q : mp){
        ss.clear();
        dfs(q.first,q.first,q.first,mp);
        for(auto g : ss) fmp[q.first].insert(g);
    }

    cout<<'\n';
    cout<<"FIRST: "<<'\n';
    for(auto q : fmp){
        string ans = "";
        ans += q.first;
        ans += " = {";
        for(char r : q.second){
            ans += r;
            ans += ',';
        }
        ans.pop_back();
        ans+="}";
        cout<<ans<<'\n';
    }

    map<char,set<char>> gmp;
    gmp[start].insert('$');
    int count = 10;
    while(count--){
        for(auto q : mp){
            for(auto r : q.second){
                for(i=0;i<r.size()-1;i++){
                    if(r[i]>='A'&&r[i]<='Z'){
                        if(!(r[i+1]>='A'&&r[i+1]<='Z')) gmp[r[i]].insert(r[i+1]);
                        else {
```

```
                    char temp = r[i+1];
                    int j = i+1;
                    while(temp>='A'&&temp<='Z'){
                        if(*fmp[temp].begin()=='e'){
                            for(auto g : fmp[temp]){
                                if(g=='e') continue;
                                gmp[r[i]].insert(g);
                            }
                            j++;
                            if(j<r.size()){
                                temp = r[j];
                                if(!(temp>='A'&&temp<='Z')){
                                    gmp[r[i]].insert(temp);
                                    break;
                                }
                            }
                            else{
                                for(auto g : gmp[q.first]) gmp[r[i]].insert(g);
                                break;
                            }
                        }
                        else{
                            for(auto g : fmp[temp]){
                                gmp[r[i]].insert(g);
                            }
                            break;
                        }
                    }
                }
            }
        }
        if(r[r.size()-1]>='A'&&r[r.size()-1]<='Z'){
            for(auto g : gmp[q.first]) gmp[r[i]].insert(g);
        }
    }
}
```

```
    }

    cout<<'\n';
    cout<<"FOLLOW: "<<'\n';
    for(auto q : gmp){
        string ans = "";
        ans += q.first;
        ans += " = {";
        for(char r : q.second){
            ans += r;
            ans += ',';
        }
        ans.pop_back();
        ans+="}";
        cout<<ans<<'\n';
    }
    return 0;
}
```

SCREENSHOTS:

output:



```
praveen@praveen-notebook:~/Desktop/SEM/CD/Lab$ co
Lab/"exp5
Grammar:
S->ACB|CbB|Ba
A->da|BC
B->g|e
C->h|e

FIRST:
A = {d,e,g,h}
B = {e,g}
C = {e,h}
S = {a,b,d,e,g,h}

FOLLOW:
A = {$,g,h}
B = {$,a,g,h}
C = {$,b,g,h}
S = {$}
```

RESULT:

The FIRST and FOLLOW set of non-terminals of a grammar were found successfully using C++

Name: Praveen Kumar K
Reg no: RA1911030010069
Subject: Compiler Design
Date: 07.03.2022

TITLE: Predictive Parser Table

AIM:

A program to implement Predictive Parser Table

ALGORITHM:

FIRST(A):

Step 1: First check for left recursion in the grammar, if there is left recursion in the grammar remove that and go to step 2.

Step 2: Calculate First() and Follow() for all non-terminals.

First(): If there is a variable, and from that variable, if we try to drive all the strings then the beginning Terminal Symbol is called the First.
Follow(): What is the Terminal Symbol which follows a variable in the process of derivation.
Step 3: For each production A –> α. (A tends to alpha)

1. Find First(α) and for each terminal in First(α), make entry A –> α in the table.

2. If First(α) contains ε (epsilon) as terminal than, find the Follow(A) and for each terminal in Follow(A), make entry A –> α in the table.

3. If the First(α) contains ε and Follow(A) contains $ as terminal, then make entry A –> α in the table for the $.

CODE:

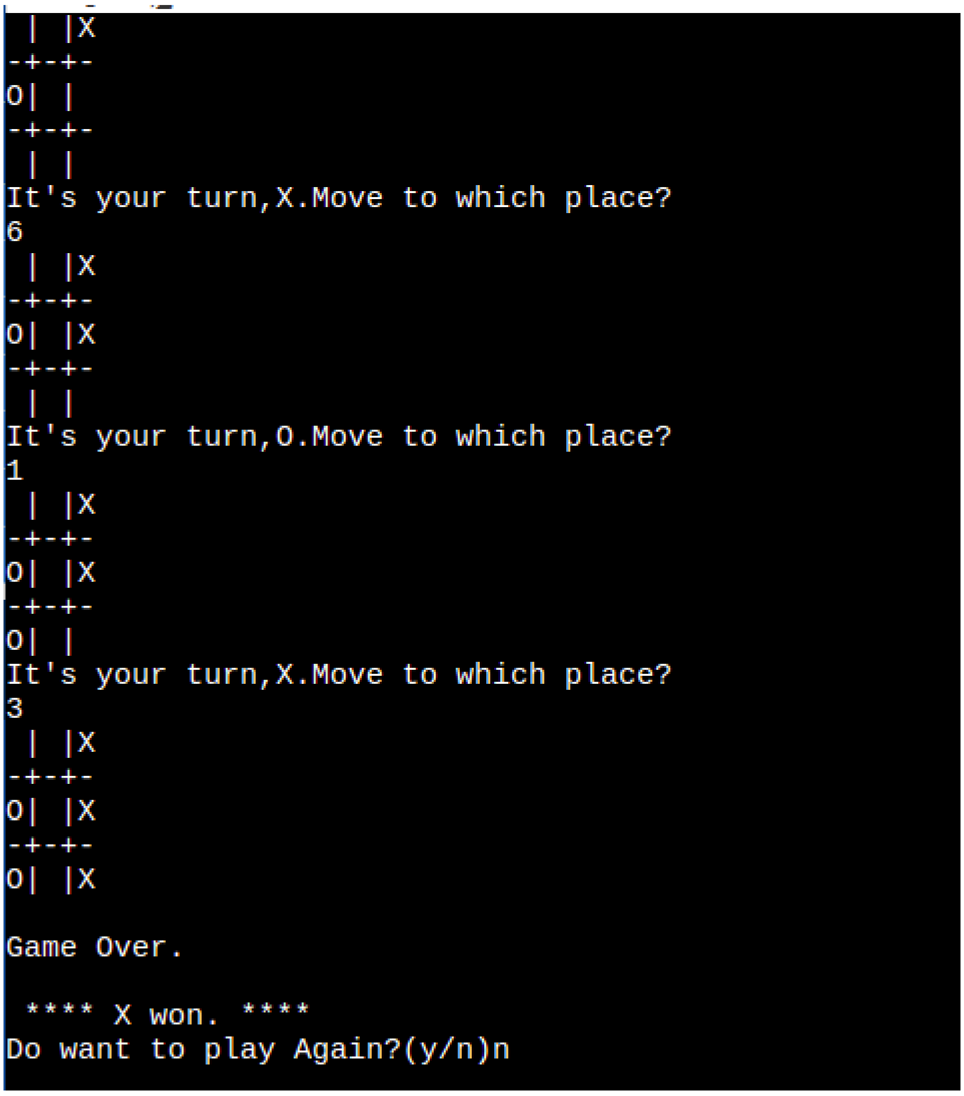
```
#include<iostream>
#include<string>
#include<deque>
using namespace std;
int n,n1,n2;
int getPosition(string arr[], string q, int size)
{
   for(int i=0;i<size;i++)
   {
      if(q == arr[i])
         return i;
   }
   return -1;
}
int main()
{
   string prods[10],first[10],follow[10],nonterms[10],terms[10];
   string pp_table[20][20] = {};
   cout<<"Enter the number of productions : ";
   cin>>n;
   cin.ignore();
   cout<<"Enter the productions"<<endl;
   for(int i=0;i<n;i++)
   {
      getline(cin,prods[i]);
      cout<<"Enter first for "<<prods[i].substr(3)<<" : ";
      getline(cin,first[i]);
   }
   cout<<"Enter the number of Terminals : ";
   cin>>n2;
   cin.ignore();
   cout<<"Enter the Terminals"<<endl;
   for(int i=0;i<n2;i++)
```

```
    {
        cin>>terms[i];
    }
    terms[n2] = "$";
    n2++;
    cout<<"Enter the number of Non-Terminals : ";
    cin>>n1;
    cin.ignore();
    for(int i=0;i<n1;i++)
    {
        cout<<"Enter Non-Terminal : ";
        getline(cin,nonterms[i]);
        cout<<"Enter follow of "<<nonterms[i]<<" : ";
        getline(cin,follow[i]);
    }



    cout<<endl;
    cout<<"Grammar"<<endl;
    for(int i=0;i<n;i++)
    {
        cout<<prods[i]<<endl;
    }



    for(int j=0;j<n;j++)
    {
        int row = getPosition(nonterms,prods[j].substr(0,1),n1);
        if(prods[j].at(3)!='#')
        {
            for(int i=0;i<first[j].length();i++)
            {
                int col = getPosition(terms,first[j].substr(i,1),n2);
                pp_table[row][col] = prods[j];
            }
        }
        else
        {
            for(int i=0;i<follow[row].length();i++)
            {
                int col = getPosition(terms,follow[row].substr(i,1),n2);
                pp_table[row][col] = prods[j];
            }
        }
    }
    //Display Table
    for(int j=0;j<n2;j++)
```

```
        cout<<"\t"<<terms[j];
    cout<<endl;
    for(int i=0;i<n1;i++)
    {
            cout<<nonterms[i]<<"\t";
            //Display Table
            for(int j=0;j<n2;j++)
            {
                cout<<pp_table[i][j]<<"\t";
            }
            cout<<endl;
    }
    //Parsing String
    char c;
    do{
    string ip;
    deque<string> pp_stack;
    pp_stack.push_front("$");
    pp_stack.push_front(prods[0].substr(0,1));
    cout<<"Enter the string to be parsed : ";
    getline(cin,ip);
    ip.push_back('$');
    cout<<"Stack\tInput\tAction"<<endl;
    while(true)
    {
        for(int i=0;i<pp_stack.size();i++)
            cout<<pp_stack[i];
        cout<<"\t"<<ip<<"\t";
        int row1 = getPosition(nonterms,pp_stack.front(),n1);
        int row2 = getPosition(terms,pp_stack.front(),n2);
        int column = getPosition(terms,ip.substr(0,1),n2);
        if(row1 != -1 && column != -1)
        {
            string p = pp_table[row1][column];
            if(p.empty())
            {
                cout<<endl<<"String cannot be Parsed."<<endl;
                break;
            }
            pp_stack.pop_front();
            if(p[3] != '#')
            {
                for(int x=p.size()-1;x>2;x--)
                {
                    pp_stack.push_front(p.substr(x,1));
                }
            }
            cout<<p;
        }
        else
        {
            if(ip.substr(0,1) == pp_stack.front())
            {
                if(pp_stack.front() == "$")
                {
                    cout<<endl<<"String Parsed."<<endl;
                    break;
                }
                cout<<"Match "<<ip[0];
                pp_stack.pop_front();
                ip = ip.substr(1);
            }
            else
            {
                cout<<endl<<"String cannot be Parsed."<<endl;
                break;
            }
        }
        cout<<endl;
    }
    cout<<"Continue?(Y/N) ";
    cin>>c;
    cin.ignore();
    }while(c=='y' || c=='Y');
    return 0;
}
```

SCREENSHOTS:

output:



```
 | |X
-+-+-
O| |
-+-+-
 | |
It's your turn,X.Move to which place?
6
 | |X
-+-+-
O| |X
-+-+-
 | |
It's your turn,O.Move to which place?
1
 | |X
-+-+-
O| |X
-+-+-
O| |
It's your turn,X.Move to which place?
3
 | |X
-+-+-
O| |X
-+-+-
O| |X

Game Over.

 **** X won. ****
Do want to play Again?(y/n)n
```

RESULT:

Predictive Parser Table has been implemented successfully using C++.

Name: Praveen Kumar K
Reg no: RA1911030010069
Subject: Compiler Design
Date: 10.03.2022

TITLE: Shift Reduce Parser

AIM:

A program to implement **Shift Reduce Parser**

ALGORITHM:

Shift Reduce parser attempts for the construction of parse in a similar manner as done in bottom-up parsing i.e. the parse tree is constructed from leaves(bottom) to the root(up). A more general form of the shift-reduce parser is the LR parser.

This parser requires some data structures i.e.

1. An input buffer for storing the input string.
2. A stack for storing and accessing the production rules.
Basic Operations –

Shift: This involves moving symbols from the input buffer onto the stack.
Reduce: If the handle appears on top of the stack then, its reduction by using appropriate production rule is done i.e. RHS of

a production rule is popped out of a stack and LHS of a production rule is pushed onto the stack.
Accept: If only the start symbol is present in the stack and the input buffer is empty then, the parsing action is called accept. When accepted action is obtained, it is means successful parsing is done.
Error: This is the situation in which the parser can neither perform shift action nor reduce action and not even accept action.

CODE:

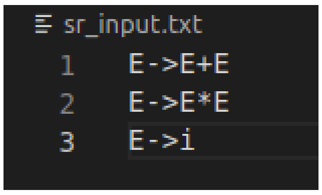
```
#include<iostream>
#include<string.h>
using namespace std;
struct prodn
{
        char p1[10];
        char p2[10];
};
int main()
{
        char input[20],stack[50],temp[50],ch[2],*t1,*t2,*t;
        int i,j,s1,s2,s,count=0;
        struct prodn p[10];
        FILE *fp=fopen("sr_input.txt","r");
        stack[0]='\0';
        cout<<"Enter the Input String:\n";
        cin>>input;
        while(!feof(fp))
        {
                fscanf(fp,"%s\n",temp);
                t1=strtok(temp,"->");
```

```
		t2=strtok(NULL,"->");
		strcpy(p[count].p1,t1);
		strcpy(p[count].p2,t2);
		count++;
	}
	i=0;
	while(1)
	{
		if(i<strlen(input))
		{
			ch[0]=input[i];
			ch[1]='\0';
			i++;
			strcat(stack,ch);
			cout<<"\n"<<stack;
		}
		for(j=0;j<count;j++)
		{
			t=strstr(stack,p[j].p2);
			if(t!=NULL)
			{
				s1=strlen(stack);
				s2=strlen(t);
				s=s1-s2;
				stack[s]='\0';
				strcat(stack,p[j].p1);
				cout<<"\n"<<stack;
				j=-1;
			}
		}
		if(strcmp(stack,"E")==0&&i==strlen(input))
		{
			cout<<"\n\nAccepted";
			break;
		}
		if(i==strlen(input))
```

```
                {
                        cout<<"\n\nNot Accepted";
                        break;
                }
        }
        return 0;
}
```

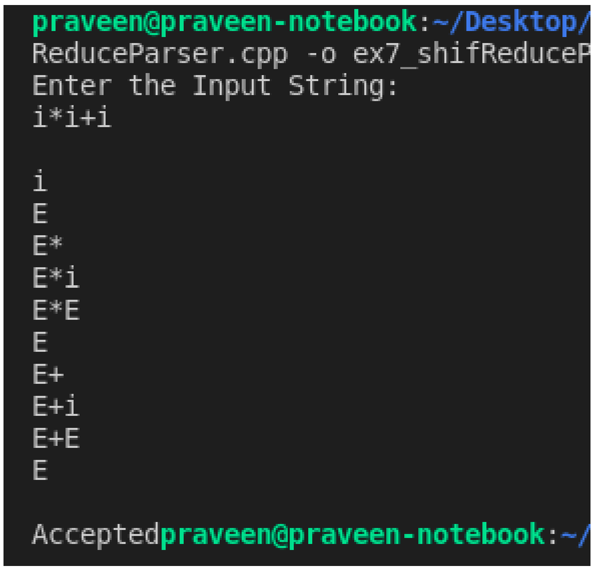
SCREENSHOTS:

input file:



```
sr_input.txt
E->E+E
E->E*E
E->i
```

output:



```
praveen@praveen-notebook:~/Desktop/
ReduceParser.cpp -o ex7_shifReduceP
Enter the Input String:
i*i+i

i
E
E*
E*i
E*E
E
E+
E+i
E+E
E

Acceptedpraveen@praveen-notebook:~/
```

RESULT:

Shift reduce parser has been implemented successfully using C++.

Name: Praveen Kumar K
Reg no: RA1911030010069
Subject: Compiler Design
Date: 01.04.2022

TITLE: Leading and Trailing

AIM :

A program to implement Leading and Trailing

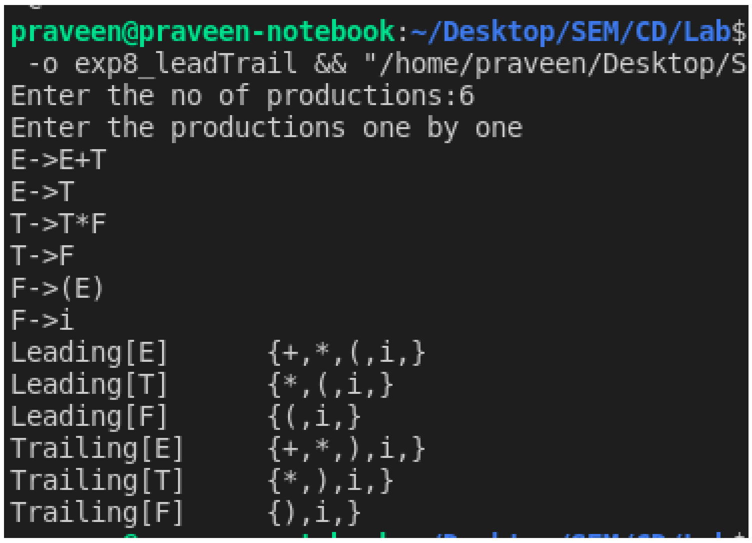
ALGORITHM :

1. For Leading, check for the first non-terminal.
2. If found, print it.
3. Look for next production for the same non-terminal.
4. If not found, recursively call the procedure for the single non-terminal present before
the
comma or End Of Production String.
5. Include it's results in the result of this non-terminal.
6. For trailing, we compute same as leading but we start from the end of the production to
the beginning.
7. Stop

CODE:

```
#include<bits/stdc++.h>

#include <cstring>
using namespace std;
int nt, t, top = 0;
char s[50], NT[10], T[10], st[50], l[10][10], tr[50][50];
int searchnt(char a)
{
   int count = -1, i;
   for (i = 0; i < nt; i++)
   {
      if (NT[i] == a)
         return i;
   }
   return count;
}
int searchter(char a)
{
   int count = -1, i;
   for (i = 0; i < t; i++)
   {
      if (T[i] == a)
         return i;
   }
   return count;
}
void push(char a)
{
   s[top] = a;
   top++;
}
char pop()
{
   top--;
   return s[top];
}
void installl(int a, int b)

{
   if (l[a][b] == 'f')
   {
      l[a][b] = 't';
      push(T[b]);
      push(NT[a]);
   }
}
void installt(int a, int b)
{
    if (tr[a][b] == 'f')
    {
        tr[a][b] = 't';
        push(T[b]);
        push(NT[a]);
    }
}

int main()
{
    int i, s, k, j, n;
    char pr[30][30], b, c;
    cout<< "Enter the no of productions:";
    cin>> n;
    cout << "Enter the productions one by one\n";
    for (i = 0; i < n; i++)
        cin >> pr[i];
    nt = 0;
    t = 0;
    for (i = 0; i < n; i++)
    {
        if ((searchnt(pr[i][0])) == -1)
            NT[nt++] = pr[i][0];
    }
    for (i = 0; i < n; i++)
    {
        for (j = 3; j < strlen(pr[i]); j++)
        {
            if (searchnt(pr[i][j]) == -1)
            {
                if (searchter(pr[i][j]) == -1)
                    T[t++] = pr[i][j];
            }
        }
    }
    for (i = 0; i < nt; i++)
    {
        for (j = 0; j < t; j++)
            l[i][j] = 'f';
    }
    for (i = 0; i < nt; i++)
    {
        for (j = 0; j < t; j++)

            tr[i][j] = 'f';
    }
    for (i = 0; i < nt; i++)
    {
        for (j = 0; j < n; j++)
        {
            if (NT[(searchnt(pr[j][0]))] == NT[i])
            {
                if (searchter(pr[j][3]) != -1)
                    installl(searchnt(pr[j][0]), searchter(pr[j][3]));
                else
                {
                    for (k = 3; k < strlen(pr[j]); k++)
                    {
                        if (searchnt(pr[j][k]) == -1)
                        {
                            installl(searchnt(pr[j][0]), searchter(pr[j][k]));
                            break;
                        }
                    }
                }
            }
        }
    }
    while (top != 0)
    {
        b = pop();
        c = pop();
        for (s = 0; s < n; s++)
        {
            if (pr[s][3] == b)
                installl(searchnt(pr[s][0]), searchter(c));
        }
    }
    for (i = 0; i < nt; i++)
    {
        cout << "Leading[" << NT[i] << "]"
             << "\t{";
        for (j = 0; j < t; j++)
        {
            if (l[i][j] == 't')
                cout << T[j] << ",";
        }
        cout << "}\n";
    }

    top = 0;
    for (i = 0; i < nt; i++)
    {
        for (j = 0; j < n; j++)
        {
            if (NT[searchnt(pr[j][0])] == NT[i])
            {
                if (searchter(pr[j][strlen(pr[j]) - 1]) != -1)
                    installt(searchnt(pr[j][0]), searchter(pr[j][strlen(pr[j]) - 1]));
                else
                {
                    for (k = (strlen(pr[j]) - 1); k >= 3; k--)
                    {
                        if (searchnt(pr[j][k]) == -1)
                        {
                            installt(searchnt(pr[j][0]), searchter(pr[j][k]));
                            break;
                        }
                    }
                }
            }
        }
    }
    while (top != 0)
    {
        b = pop();
        c = pop();
        for (s = 0; s < n; s++)
        {
            if (pr[s][3] == b)
                installt(searchnt(pr[s][0]), searchter(c));
        }
    }
    for (i = 0; i < nt; i++)
    {
        cout << "Trailing[" << NT[i] << "]"
             << "\t{";
        for (j = 0; j < t; j++)
        {
            if (tr[i][j] == 't')
                cout << T[j] << ",";
        }
        cout << "}\n";
    }
    return 0;
}
```

SCREENSHOTS:



```
praveen@praveen-notebook:~/Desktop/SEM/CD/Lab$
 -o exp8_leadTrail && "/home/praveen/Desktop/S
Enter the no of productions:6
Enter the productions one by one
E->E+T
E->T
T->T*F
T->F
F->(E)
F->i
Leading[E]      {+,*,(,i,}
Leading[T]      {*,(,i,}
Leading[F]      {(,i,}
Trailing[E]     {+,*,),i,}
Trailing[T]     {*,),i,}
Trailing[F]     {),i,}
```

RESULT:

The program was successfully compiled and run using c++.

Name: Praveen Kumar K
Reg no: RA1911030010069
Subject: Compiler Design
Date: 02.04.2022

TITLE: LR(0)

Aim:

A program to implement LR(0) items

Algorithm:-

1. Start.
2. Create structure for production with LHS and RHS.
3. Open file and read input from file.
4. Build state 0 from extra grammar Law S' -> S $ that is all start symbol of grammar and one
Dot (.) before S symbol.
5. If Dot symbol is before a non-terminal, add grammar laws that this non-terminal is in Left
Hand Side of that Law and set Dot in before of first part of Right Hand Side.
6. If state exists (a state with this Laws and same Dot position), use that instead.
7. Now find set of terminals and non-terminals in which Dot exist in before.
8. If step 7 Set is non-empty go to 9, else go to 10.

9. For each terminal/non-terminal in set step 7 create new state by using all grammar law that
Dot position is before of that terminal/non-terminal in reference state by increasing Dot point
to next part in Right Hand Side of that laws.
10. Go to step 5.
11. End of state building.
12. Display the output.
13. End.

CODE:

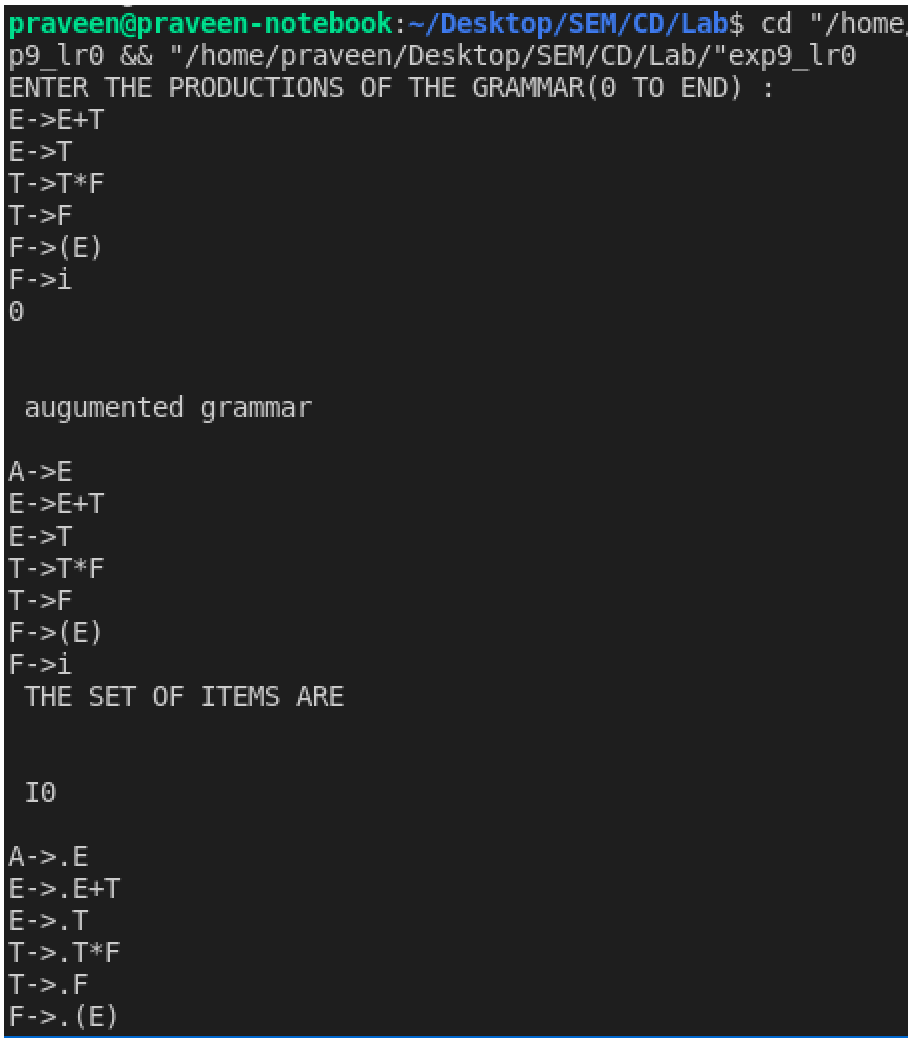
```
#include<iostream>
#include<cstring>

using namespace std;

char prod[20][20],listofvar[26]="ABCDEFGHIJKLMNOPQR";
int novar=1,i=0,j=0,k=0,n=0,m=0,arr[30];
int noitem=0;

struct Grammar
{
	char lhs;
	char rhs[8];
}g[20],item[20],clos[20][10];

int isvariable(char variable)
{
	for(int i=0;i<novar;i++)
		if(g[i].lhs==variable)
			return i+1;
	return 0;
}
```

```
void findclosure(int z, char a)
{
	int n=0,i=0,j=0,k=0,l=0;
	for(i=0;i<arr[z];i++)
	{
		for(j=0;j<strlen(clos[z][i].rhs);j++)
		{
			if(clos[z][i].rhs[j]=='.' && clos[z][i].rhs[j+1]==a)
			{
				clos[noitem][n].lhs=clos[z][i].lhs;
				strcpy(clos[noitem][n].rhs,clos[z][i].rhs);
				char temp=clos[noitem][n].rhs[j];
				clos[noitem][n].rhs[j]=clos[noitem][n].rhs[j+1];
				clos[noitem][n].rhs[j+1]=temp;
				n=n+1;
			}
		}
	}
	for(i=0;i<n;i++)
	{
		for(j=0;j<strlen(clos[noitem][i].rhs);j++)
		{
			if(clos[noitem][i].rhs[j]=='.' && isvariable(clos[noitem]
[i].rhs[j+1])>0)
			{
				for(k=0;k<novar;k++)
				{
					if(clos[noitem][i].rhs[j+1]==clos[0][k].lhs)
					{
						for(l=0;l<n;l++)
							if(clos[noitem][l].lhs==clos[0][k].lhs
&& strcmp(clos[noitem][l].rhs,clos[0][k].rhs)==0)
								break;
						if(l==n)
						{
							clos[noitem][n].lhs=clos[0][k].lhs;
```

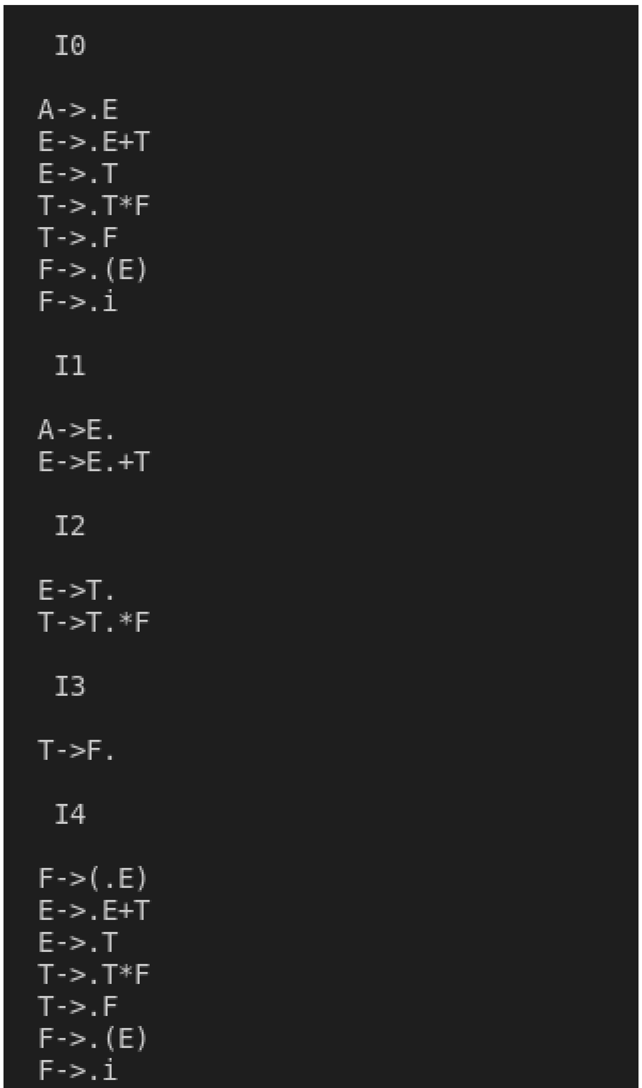
```
                                                            strcpy(clos[noitem][n].rhs,clos[0][k].rhs);
                                                                      n=n+1;
                                                            }
                                                  }
                                        }
                              }
                    }
          }
          arr[noitem]=n;
          int flag=0;
          for(i=0;i<noitem;i++)
          {
                    if(arr[i]==n)
                    {
                              for(j=0;j<arr[i];j++)
                              {
                                        int c=0;
                                        for(k=0;k<arr[i];k++)
                                                  if(clos[noitem][k].lhs==clos[i][k].lhs &&
strcmp(clos[noitem][k].rhs,clos[i][k].rhs)==0)
                                                            c=c+1;
                                        if(c==arr[i])
                                        {
                                                  flag=1;
                                                  goto exit;
                                        }
                              }
                    }
          }
          exit:;
          if(flag==0)
                    arr[noitem++]=n;
}

int main()
{
	cout<<"ENTER THE PRODUCTIONS OF THE GRAMMAR(0 TO END) :\n";
	do
	{
		cin>>prod[i++];
	}while(strcmp(prod[i-1],"0")!=0);
	for(n=0;n<i-1;n++)
	{
		m=0;
		j=novar;
		g[novar++].lhs=prod[n][0];
		for(k=3;k<strlen(prod[n]);k++)
		{
			if(prod[n][k] != '|')
			g[j].rhs[m++]=prod[n][k];
			if(prod[n][k]=='|')
			{
				g[j].rhs[m]='\0';
				m=0;
				j=novar;
				g[novar++].lhs=prod[n][0];
			}
		}
	}
	for(i=0;i<26;i++)
		if(!isvariable(listofvar[i]))
			break;
	g[0].lhs=listofvar[i];
	char temp[2]={g[1].lhs,'\0'};
	strcat(g[0].rhs,temp);
	cout<<"\n\n augumented grammar \n";
	for(i=0;i<novar;i++)
		cout<<endl<<g[i].lhs<<"->"<<g[i].rhs<<" ";

	for(i=0;i<novar;i++)
	{
		clos[noitem][i].lhs=g[i].lhs;
```

```
                strcpy(clos[noitem][i].rhs,g[i].rhs);
                if(strcmp(clos[noitem][i].rhs,"ε")==0)
                        strcpy(clos[noitem][i].rhs,".");
                else
                {
                        for(int j=strlen(clos[noitem][i].rhs)+1;j>=0;j--)
                                clos[noitem][i].rhs[j]=clos[noitem][i].rhs[j-1];
                        clos[noitem][i].rhs[0]='.';
                }
        }
        arr[noitem++]=novar;
        for(int z=0;z<noitem;z++)
        {
                char list[10];
                int l=0;
                for(j=0;j<arr[z];j++)
                {
                        for(k=0;k<strlen(clos[z][j].rhs)-1;k++)
                        {
                                if(clos[z][j].rhs[k]=='.')
                                {
                                        for(m=0;m<l;m++)
                                                if(list[m]==clos[z][j].rhs[k+1])
                                                        break;
                                        if(m==l)
                                                list[l++]=clos[z][j].rhs[k+1];
                                }
                        }
                }
                for(int x=0;x<l;x++)
                        findclosure(z,list[x]);
        }
        cout<<"\n THE SET OF ITEMS ARE \n\n";
        for(int z=0; z<noitem; z++)
        {
                cout<<"\n I"<<z<<"\n\n";
                for(j=0;j<arr[z];j++)
                        cout<<clos[z][j].lhs<<"->"<<clos[z][j].rhs<<"\n";

        }

}
```

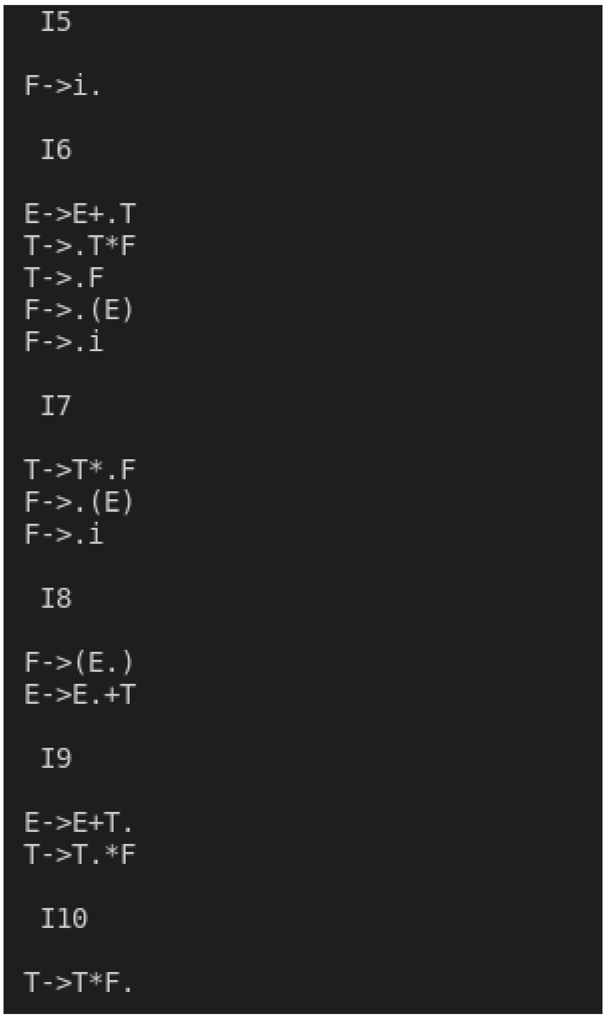
SCREENSHOTS:



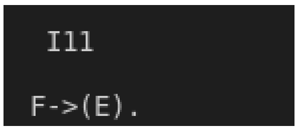
```
praveen@praveen-notebook:~/Desktop/SEM/CD/Lab$ cd "/home
p9_lr0 && "/home/praveen/Desktop/SEM/CD/Lab/"exp9_lr0
ENTER THE PRODUCTIONS OF THE GRAMMAR(0 TO END) :
E->E+T
E->T
T->T*F
T->F
F->(E)
F->i
0


 augumented grammar

A->E
E->E+T
E->T
T->T*F
T->F
F->(E)
F->i
 THE SET OF ITEMS ARE


 I0

A->.E
E->.E+T
E->.T
T->.T*F
T->.F
F->.(E)
```



```
 I0

A->.E
E->.E+T
E->.T
T->.T*F
T->.F
F->.(E)
F->.i

 I1

A->E.
E->E.+T

 I2

E->T.
T->T.*F

 I3

T->F.

 I4

F->(.E)
E->.E+T
E->.T
T->.T*F
T->.F
F->.(E)
F->.i
```



```
 I5

F->i.

 I6

E->E+.T
T->.T*F
T->.F
F->.(E)
F->.i

 I7

T->T*.F
F->.(E)
F->.i

 I8

F->(E.)
E->E.+T

 I9

E->E+T.
T->T.*F

 I10

T->T*F.
```



```
 I11
F->(E).
```

RESULT:

The program was successfully compiled and run.

Name: Praveen Kumar K
Reg no: RA1911030010069
Subject: Compiler Design
Date: 02.04.2022

TITLE: Postfix and Prefix

Aim:
A program to implement Intermediate code generation – Postfix, Prefix.

Algorithm:-

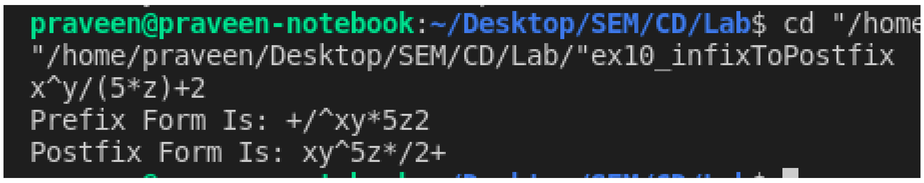
1. Declare set of operators.
2. Initialize an empty stack.
3. To convert INFIX to POSTFIX follow the following steps
4. Scan the infix expression from left to right.
5. If the scanned character is an operand, output it.
6. Else, If the precedence of the scanned operator is greater than the precedence of the
operator in the stack(or the stack is empty or the stack contains a '('), push it.
7. Else, Pop all the operators from the stack which are greater than or equal to in precedence
than that of the scanned operator. After doing that Push the scanned operator to the stack.
8. If the scanned character is an '(', push it to the stack.

9. If the scanned character is an ‘)’, pop the stack and output it until a ‘(‘ is encountered, and
discard both the parenthesis.
10. Pop and output from the stack until it is not empty.
11. To convert INFIX to PREFIX follow the following steps
12. First, reverse the infix expression given in the problem.
13. Scan the expression from left to right.
14. Whenever the operands arrive, print them.
15. If the operator arrives and the stack is found to be empty, then simply push the operator
into the stack.
16. Repeat steps 6 to 9 until the stack is empty

CODE:
```
#include<bits/stdc++.h>
using namespace std;
int preced(char ch) {
   if(ch == '+' || ch == '-') {
      return 1;    //Precedence of + or - is 1
   }else if(ch == '*' || ch == '/') {
      return 2;    //Precedence of * or / is 2
   }else if(ch == '^') {
      return 3;    //Precedence of ^ is 3
   }else {
      return 0;
   }
}

string inToPost(string infix) {
   stack<char> stk;
   stk.push('#');    //add some extra character to avoid underflow
   string postfix = "";   //initially the postfix string is empty
   string::iterator it;

   for(it = infix.begin(); it!=infix.end(); it++) {
      if(isalnum(char(*it)))
         postfix += *it;    //add to postfix when character is letter or number
```

```
    else if(*it == '(')
      stk.push('(');
    else if(*it == '^')
      stk.push('^');
    else if(*it == ')') {
      while(stk.top() != '#' && stk.top() != '(') {
        postfix += stk.top();    //store and pop until ( has found
        stk.pop();
      }

      stk.pop();    //remove the '(' from stack
    }else {
      if(preced(*it) > preced(stk.top()))
        stk.push(*it);    //push if precedence is high
      else {
        while(stk.top() != '#' && preced(*it) <= preced(stk.top())) {
          postfix += stk.top();    //store and pop until higher precedence is found
          stk.pop();
        }
        stk.push(*it);
      }
    }
  }

  while(stk.top() != '#') {
    postfix += stk.top();    //store and pop until stack is not empty

    stk.pop();

  }
  return postfix;
}

string inToPre(string infix) {
  string prefix;
  reverse(infix.begin(), infix.end());    //reverse the infix expression
  string::iterator it;

  for(it = infix.begin(); it != infix.end(); it++) {    //reverse the parenthesis after reverse
    if(*it == '(')
      *it = ')';
    else if(*it == ')')
      *it = '(';
  }

  prefix = inToPost(infix);                //convert new reversed infix to postfix form.
  reverse(prefix.begin(), prefix.end());    //again reverse the result to get final prefix form
  return prefix;
}

int main()
{
   string infix;
   cin>>infix;
   cout << "Prefix Form Is: " << inToPre(infix) << endl;
   cout << "Postfix Form Is: " << inToPost(infix) << endl;


         return 0;
}
```

SCREENSHOTS:



```
praveen@praveen-notebook:~/Desktop/SEM/CD/Lab$ cd "/home
"/home/praveen/Desktop/SEM/CD/Lab/"ex10_infixToPostfix
x^y/(5*z)+2
Prefix Form Is: +/^xy*5z2
Postfix Form Is: xy^5z*/2+
```

RESULT:

Infix to Prefix and Infix to Postfix has been performed successfully.

Name: Praveen Kumar K
Reg no: RA1911030010069
Subject: Compiler Design
Date: 10.04.2022

TITLE: Implementation of Intermediate code generation – Quadruple, Triple, Indirect triple

Aim:

Intermediate code generation – Quadruple, Triple, Indirect triple

Algorithm:

The algorithm takes a sequence of three-address statements as input. For each three address
statements of the form a:= b op c perform the various actions. These are as follows:

1. Invoke a function getreg to find out the location L where the result of computation b op c should be stored.

2. Consult the address description for y to determine y'. If the value of y currently in memory and register both then prefer the register y' . If the value of y is not already in L then generate the instruction MOV y' , L to place a copy of y in L.

3. Generate the instruction OP z' , L where z' is used to show the current location of z. if z is in both then prefer a register to a memory location. Update the address descriptor of x to

indicate that x is in location L. If x is in L then update its descriptor and remove x from all other descriptors.

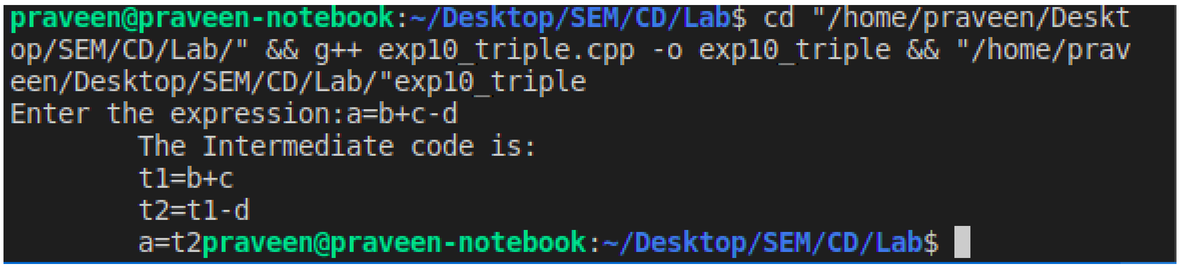
4. If the current value of y or z have no next uses or not live on exit from the block or in register then alter the register descriptor to indicate that after execution of x : = y op z those register will no longer contain y or z.

CODE:

```
#include<stdio.h>

#include<ctype.h>#include<stdlib.h>

#include<string.h>

void small();
void dove(int i);
int p[5] = {
  0,
  1,
  2,
  3,
  4
}, c = 1, i, k, l, m, pi;
char sw[5] = {
  '=',
  '-',
  '+',
  '/',
  '*'
```

```
}, j[20], a[5], b[5], ch[2];
void main() {
  printf("Enter the expression:");
  scanf("%s", j);
  printf("\tThe Intermediate code is:\n");
  small();
}
void dove(int i) {
  a[0] = b[0] = '\0';
  if (!isdigit(j[i + 2]) && !isdigit(j[i - 2])) {
    a[0] = j[i - 1];
    b[0] = j[i + 1];
  }
  if (isdigit(j[i + 2])) {
    a[0] = j[i - 1];
    b[0] = 't';
    b[1] = j[i + 2];
  }
  if (isdigit(j[i - 2])) {
    b[0] = j[i + 1];
    a[0] = 't';
    a[1] = j[i - 2];
    b[1] = '\0';
  }
  if (isdigit(j[i + 2]) && isdigit(j[i - 2])) {
    a[0] = 't';
    b[0] = 't';
    a[1] = j[i - 2];
    b[1] = j[i + 2];
    sprintf(ch, "%d", c);
    j[i + 2] = j[i - 2] = ch[0];
  }
  if (j[i] == '*')
    printf("\tt%d=%s*%s\n", c, a, b);
  if (j[i] == '/') printf("\tt%d=%s/%s\n", c, a, b);
  if (j[i] == '+')
    printf("\tt%d=%s+%s\n", c, a, b);
  if (j[i] == '-')
    printf("\tt%d=%s-%s\n", c, a, b);
  if (j[i] == '=')
    printf("\t%c=t%d", j[i - 1], --c);
  sprintf(ch, "%d", c);
  j[i] = ch[0];
  c++;
  small();
}
void small() {
  pi = 0;
  l = 0;
  for (i = 0; i < strlen(j); i++) {
    for (m = 0; m < 5; m++)
      if (j[i] == sw[m])
        if (pi <= p[m]) {
          pi = p[m];
          l = 1;
          k = i;
        }
  }
  if (l == 1)
    dove(k);
  else
    exit(0);
}
```

SCREENSHOTS:



```
praveen@praveen-notebook:~/Desktop/SEM/CD/Lab$ cd "/home/praveen/Deskt
op/SEM/CD/Lab/" && g++ exp10_triple.cpp -o exp10_triple && "/home/prav
een/Desktop/SEM/CD/Lab/"exp10_triple
Enter the expression:a=b+c-d
        The Intermediate code is:
        t1=b+c
        t2=t1-d
        a=t2praveen@praveen-notebook:~/Desktop/SEM/CD/Lab$
```

RESULT:

The program was successfully compiles using c++.