Program No. 1

Aim :- Implement the idea of lexical analyzer.

Source Code :-

```
/*
  Lab Name - Compiler Design
  Objective - To implement the idea of Lexical Analysis
  Name - Anup Agrawal
  Roll No. - UE143014
  Date -  18/01/2017 ,25/01/2017 & 01/02/2017
*/
#include<iostream>
#include<fstream>
#include<string.h>
#include<sstream>
#include<vector>
#include<algorithm>
#define SIZE 100

using namespace std;

int main()
{
   ifstream in,key,op;
   ofstream out;
   in.open("third.txt",ios::in);
   key.open("keywords.txt",ios::in);
   op.open("operators.txt",ios::in);
   out.open("output.txt");
   int line = 0;
   char ch_in,ch_key,ch_op;
   ifstream in1;
   ofstream out1;
   char ch;
   int Line = 1,comment=0;
   cout<<"1. ";
   in1.open("input.txt",ios::in);
   out1.open("temp.txt");
```

```
while(!in1.eof())
{
   in1.get(ch);
   switch(ch)
   {
 case "":
   {
      cout<<ch;
      in1.get(ch);
      while(ch !="")
      {
          cout<<ch;
          in1.get(ch);
      }
      cout<<"";
        break;
   }
 case '/' :
   in1.get(ch);
   if(ch == '/')
   {
     comment++;
     while(in1.get(ch))
     {
         if(ch == '\n')
          {
           cout<<ch;
           Line++;
           cout<<Line<<". ";
           break;
          }
     }
   }
     else if(ch == '*')
     {
      comment++;
      in1.get(ch);
      while(in1.get(ch))
      {
```

```
            if(ch == '\n')
            {
             cout<<ch;
             Line++;
             cout<<Line<<". ";
            }
            if(ch == '*')
            {
               in1.get(ch);
               if(ch == '/')
               {
                  break;
               }
            }
        }
      }
      else{
        cout<<'/';
        cout<<ch;
        out1.put('/');
        out1.put(ch);
      }
      break;

  default:
     cout<<ch;
      if(ch == '\n')
     {
        Line++;
        cout<<Line<<". ";
     }
}
}
cout<<endl<<"******"<<comment<<"  comments removed successfully *********"<<endl;
in1.close();
out1.close();
vector<string> v;
vector<string> keyw;
vector<string> ope;
```

```
 vector<string> fkeyw;
 vector<string> fope;
 string word;
 string temp = "";
 vector<int> li;
 vector<int> fli;

 while(!in.eof())
 {
  in.get(ch_in);
  if(ch_in == '\n')
  {
     line++;
  }
    if(ch_in !=';' && ch_in != '(' && ch_in !=' ' && ch_in != '\n' && ch_in!=')' && ch_in != '}'
&& ch_in != '{' && ch_in !='<' && ch_in != '>' && ch_in !='.' && ch_in != '#')
     {
        if(ch_in != '\n')
        {
          temp += ch_in;
        }
     }
     else{
       if(temp != "")
       {
          v.push_back(temp);
          li.push_back(line);
       }
       temp ="";
     }
  }
 v.push_back("}");
 li.push_back(line);
 while(getline(key,word))
 {
    keyw.push_back(word);
 }
 while(getline(op,word))
 {
```

```
    ope.push_back(word);
}
cout<<" ********** Detected Keywords ***********"<<endl;
out<<" ********** Detected Keywords ***********"<<endl;
for(int i=0;i<v.size();i++)
{
    for(int j=0;j<keyw.size();j++)
    {
        if(v[i] == keyw[j])
        {
            fkeyw.push_back(v[i]);
            cout<<"Line Number ---> "<<li[i]<<" --->"<<v[i]<<" ----> "<<" Keyword"<<endl;
            out<<"Line Number ---> "<<li[i]<<" --->"<<v[i]<<endl;
        }
    }
}
cout<<" ********** Detected Operators ***********"<<endl;
out<<" ********** Detected Operators ***********"<<endl;
for(int i=0;i<v.size();i++)
{
    for(int j=0;j<ope.size();j++)
    {
        if(v[i] == ope[j])
        {
            fope.push_back(v[i]);
            cout<<"Line Number ---> "<<li[i]<<" ---> "<<v[i]<<" ----> "<<" Operator"<<endl;
            out<<"Line Number ---> "<<li[i]<<" ---> "<<v[i]<<endl;
        }
    }
}
cout<<"************* Identifiers **********************"<<endl;
out<<"************* Identifiers **********************"<<endl;
vector<string> res;
vector<string> fres;
set_symmetric_difference(v.begin(), v.end(),fkeyw.begin(), fkeyw.end(),back_inserter(res));
int pos1;
fres = v;
for(int i=0;i<fkeyw.size();i++)
{
```

```
      pos1 = (find(fres.begin(),fres.end(),fkeyw[i])) - fres.begin();
      fres[pos1] = "";
   }
   for(int i=0;i<fres.size();i++)
   {
      if(fres[i] == "")
      {
         fres.erase(fres.begin() + i);
         i = i - 1;
      }
   }
   int pos;
   for(int i=0;i<fres.size();i++)
   {
    pos = (find(v.begin(),v.end(),fres[i])) - v.begin();
    string s = fres[i];
    int length;
    length = s.length();
    bool flag = true;
    int asc = (int)s[0];
    if(!((asc >= 65 && asc <= 90) || (asc >=97 && asc <=122)))
    {
      flag = false;
      cout<<"Invalid Identifier : "<< s <<" ,Error : Identifier must start with an alpha at Line ---->"
<<li[pos]<<endl;
      out<<"Invalid Identifier : "<< s <<" ,Error : Identifier must start with an alpha at Line ----> "
<<li[pos]<<endl;
    }
    else{

       for(int j=1;j < (length - 1);j++)
       {
          asc = (int) s[j];
          if(!((asc >= 65 && asc <= 90) || (asc >=97 && asc <=122) || (asc >= 48 && asc <= 57) ||
(asc == 95)))
          {
             flag = false;
             cout<<"Invalid Identifier : "<< s <<" ,Error : Identifier cannot contain " <<s[j]<<" at
Line ----> "<<li[pos]<<endl;
```

```
            out<<"Invalid Identifier : "<< s <<" ,Error : Identifier cannot contain " <<s[j]<<" at
Line ----> "<<li[pos]<<endl;
        }
    }

        asc = (int) s[length -1];
        if(!((asc >= 65 && asc <= 90) || (asc >=97 && asc <=122) || (asc >= 48 && asc <= 57) ||
(asc == 95) || (asc == 42)))
        {
            flag = false;
            cout<<"Invalid Identifier : "<< s <<" ,Error : Identifier cannot contain " <<s[length -
1]<<" at Line ----> "<<li[pos]<<endl;
            out<<"Invalid Identifier : "<< s <<" ,Error : Identifier cannot contain " <<s[length -
1]<<" at Line ----> "<<li[pos]<<endl;
        }
  }
  if(flag)
  {
    cout<<"Valid Identifier in Line ----> "<<li[pos]<<" -----> "<<s<<endl<<endl;
    out<<"Valid Identifier in Line ----> "<<li[pos]<<" -----> "<<s<<endl<<endl;
  }
  else{
    cout<<"Invalid Identifier in Line ----> "<<li[pos]<<"----->"<<s<<endl<<endl;
    out<<"Invalid Identifier in Line ----> "<<li[pos]<<"----->"<<s<<endl<<endl;
  }
 }
 in.close();
 key.close();
 op.close();
 out.close();
 return 0;
}
```

Contents of file third.txt :

```
#include<iostream>

int main()
{
int ak;
```

```
int anup;
int abc_19;
int abc19;
int abc_;
int abc_abc;
int abc__;
int abc_19__2017_CD;
int abc_19_;
int abc$;
int abc#;
int &abc;
int abs#x;
int abv@_19;
int abc*;
if(a == 4)
{
cout<<"Right";
}
else{
cout<<"Wrong";
}
for(int i = 6;i < 10; i++)
{
while( j + a != 15)
{
int c = k + l;
}
}
return 0;
}
```

Contents of file input.txt :

```
#include <iostream>

using namespace std;

int main()
{
```

```
Anup
Anup

a = b/a;
cout << " // CD ";
//------
// -----Compiler Design
//----Comments
------
/* aaaaaaaaa
a*/
retrun 0;

}
```

Keyword.txt contains keywords.
Operator.txt contains operators.

Contents of output.txt :

```
********** Detected Keywords ***********
Line Number ---> 0 --->include
Line Number ---> 0 --->iostream
Line Number ---> 2 --->int
Line Number ---> 2 --->main
Line Number ---> 4 --->int
Line Number ---> 5 --->int
Line Number ---> 6 --->int
Line Number ---> 7 --->int
Line Number ---> 8 --->int
Line Number ---> 9 --->int
Line Number ---> 10 --->int
Line Number ---> 11 --->int
Line Number ---> 12 --->int
Line Number ---> 13 --->int
Line Number ---> 14 --->int
Line Number ---> 15 --->int
Line Number ---> 16 --->int
Line Number ---> 17 --->int
```

Line Number ---> 18 --->int
Line Number ---> 19 --->if
Line Number ---> 21 --->cout
Line Number ---> 23 --->else
Line Number ---> 24 --->cout
Line Number ---> 26 --->for
Line Number ---> 26 --->int
Line Number ---> 28 --->while
Line Number ---> 30 --->int
Line Number ---> 33 --->return
********** Detected Operators **********
Line Number ---> 19 ---> ==
Line Number ---> 26 ---> =
Line Number ---> 28 ---> +
Line Number ---> 28 ---> !=
Line Number ---> 30 ---> =
Line Number ---> 30 ---> +
********** Identifiers **********
Valid Identifier in Line ----> 4 -----> ak

Valid Identifier in Line ----> 5 -----> anup

Valid Identifier in Line ----> 6 -----> abc_19

Valid Identifier in Line ----> 7 -----> abc19

Valid Identifier in Line ----> 8 -----> abc_

Valid Identifier in Line ----> 9 -----> abc_abc

Valid Identifier in Line ----> 10 -----> abc__

Valid Identifier in Line ----> 11 -----> abc_19__2017_CD

Valid Identifier in Line ----> 12 -----> abc_19_

Invalid Identifier : abc$,Error : Identifier cannot contain $ at Line ----> 13
Invalid Identifier in Line ----> 13----->abc$

Valid Identifier in Line ----> 14 -----> abc

Invalid Identifier : &abc ,Error : Identifier must start with an alpha at Line ----> 15
Invalid Identifier in Line ----> 15----->&abc

Valid Identifier in Line ----> 16 -----> abs

Valid Identifier in Line ----> 16 -----> x

Invalid Identifier : abv@_19 ,Error : Identifier cannot contain @ at Line ----> 17
Invalid Identifier in Line ----> 17----->abv@_19

Valid Identifier in Line ----> 18 -----> abc*

Valid Identifier in Line ----> 19 -----> a

Invalid Identifier : == ,Error : Identifier must start with an alpha at Line ----> 19
Invalid Identifier in Line ----> 19----->==

Invalid Identifier : 4 ,Error : Identifier must start with an alpha at Line ----> 19
Invalid Identifier in Line ----> 19----->4

Invalid Identifier : "Right" ,Error : Identifier must start with an alpha at Line ----> 21
Invalid Identifier in Line ----> 21----->"Right"

Invalid Identifier : "Wrong" ,Error : Identifier must start with an alpha at Line ----> 24
Invalid Identifier in Line ----> 24----->"Wrong"

Valid Identifier in Line ----> 26 -----> i

Invalid Identifier : = ,Error : Identifier must start with an alpha at Line ----> 26
Invalid Identifier in Line ----> 26----->=

Invalid Identifier : 6 ,Error : Identifier must start with an alpha at Line ----> 26
Invalid Identifier in Line ----> 26----->6

Valid Identifier in Line ----> 26 -----> i

Invalid Identifier : 10 ,Error : Identifier must start with an alpha at Line ----> 26

Invalid Identifier in Line ----> 26----->10

Invalid Identifier : i++ ,Error : Identifier cannot contain + at Line ----> 26
Invalid Identifier : i++ ,Error : Identifier cannot contain + at Line ----> 26
Invalid Identifier in Line ----> 26----->i++

Valid Identifier in Line ----> 28 -----> j

Invalid Identifier : + ,Error : Identifier must start with an alpha at Line ----> 28
Invalid Identifier in Line ----> 28----->+

Valid Identifier in Line ----> 19 -----> a

Invalid Identifier : != ,Error : Identifier must start with an alpha at Line ----> 28
Invalid Identifier in Line ----> 28----->!=

Invalid Identifier : 15 ,Error : Identifier must start with an alpha at Line ----> 28
Invalid Identifier in Line ----> 28----->15

Valid Identifier in Line ----> 30 -----> c

Invalid Identifier : = ,Error : Identifier must start with an alpha at Line ----> 26
Invalid Identifier in Line ----> 26----->=

Valid Identifier in Line ----> 30 -----> k

Invalid Identifier : + ,Error : Identifier must start with an alpha at Line ----> 28
Invalid Identifier in Line ----> 28----->+

Valid Identifier in Line ----> 30 -----> l

Invalid Identifier : 0 ,Error : Identifier must start with an alpha at Line ----> 33
Invalid Identifier in Line ----> 33----->0

Invalid Identifier : } ,Error : Identifier must start with an alpha at Line ----> 34
Invalid Identifier in Line ----> 34----->}

Removed Comments from input.txt :

```
"F:\Semester 6\Labs\Compiler_Design\CD_Program_1.exe"
1. #include <iostream>
2.
3. using namespace std;
4.
5. int main()
6. {
7. Anup
8. Anup
9.
10. a = b/a;
11. cout << " // CD ";
12.
13.
14.
15. ------
16.
17.
18. retrun 0;
19.
20. }}
******4  comments removed successfully *********
```

Program No. 2

Aim : Program to convert a regular expression into finite automata.

Source Code :

```
/*
  Lab Name - Compiler Design
  Objective - Coversion of regular expression into finite automata
  Name - Anup Agrawal
  Roll No. - UE143014
  Date - 08/02/2017
*/

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

vector< vector<string> > table;
int state = 0;

string ConvertIntoString(int number)
{
   stringstream ss ;
         ss << number ;
         return ss.str();
}

bool IsOperand(char C)
{
         if(C == 'a' || C == 'b')
   {
      return true;
   }
   else
   {
      return false;
   }
}

bool IsOperator(char C)
```

```
{
        if(C == '|' || C == '.')
   {
      return true;
   }
   else
   {
      return false;
   }
}

int GetOperatorWeight(char op)
{
        int weight = -1;
        switch(op)
        {
        case '|':
                weight = 10;
                break;
        case '.':
                weight = 20;
                break;
        }
        return weight;
}

bool HasHigherPrecedence(char op1, char op2)
{
        int op1Weight = GetOperatorWeight(op1);
        int op2Weight = GetOperatorWeight(op2);
        return ((op1Weight > op2Weight) ?  true : false);
}

string ConvertIntoPostfix(string re)
{
        stack<char> S;
        string postfix = "";
        for(int i = 0;i<re.length();i++)
      {
```

```
    if(re[i] == ' ' || re[i] == ',') continue;
    else if(IsOperator(re[i]))
            {
               if(!S.empty()&& HasHigherPrecedence('.','|'))
               {
                  postfix += S.top();
                  S.pop();
               }
               S.push(re[i]);
            }
    else if(re[i] == '*')
    {
       postfix+= re[i];
    }
    else if(IsOperand(re[i]))
            {
               postfix +=re[i];
            }

    else if (re[i] == '(')
            {
               S.push(re[i]);
            }

    else if(re[i] == ')')
            {
               while(!S.empty() && S.top() !=  '(')
                     {
                        postfix += S.top();
                        S.pop();
                     }
            S.pop();
            }
}

    while(!S.empty())
{
    postfix += S.top();
    S.pop();
```

```
    }
  cout<< "Postfix Expression for Entered Regular Expression ----> "<<postfix<<endl;
        return postfix;
}

void Only_a_b(int &in , int &fi , char ch) {
        vector<string> temp ;
        if(state == 0)
  {
        in = state ;
        fi = state + 1 ;
        state += 1 ;
  }
  else{
  in = state + 1;
  fi = state + 2;
  state += 2;
  }
        temp . push_back(ConvertIntoString(in)) ;
        temp . push_back(ConvertIntoString(fi)) ;
        temp . push_back(string(1 , ch)) ;
        table . push_back(temp) ;
        temp . clear() ;
}

void ReConcatenationRe(int &in , int &fi , int in1 , int fi1 , int in2 , int fi2) {
        in = in1 ;
        fi = fi2 ;
        vector<string> temp ;
        temp . push_back(ConvertIntoString(fi1)) ;
        temp . push_back(ConvertIntoString(in2)) ;
        temp . push_back("E") ;
        table . push_back(temp) ;
        temp . clear() ;
}

void ReUnionRe(int &in , int &fi , int in1 , int fi1 , int in2 , int fi2) {
        in = state + 1 ;
        fi = state + 2 ;
```

```
        state += 2 ;
        vector<string> temp ;
        temp . push_back(ConvertIntoString(in)) ;
        temp . push_back(ConvertIntoString(in1)) ;
        temp . push_back("E") ;
        table . push_back(temp) ;
        temp . clear() ;
        temp . push_back(ConvertIntoString(in)) ;
        temp . push_back(ConvertIntoString(in2)) ;
        temp . push_back("E") ;
        table . push_back(temp) ;
        temp . clear() ;
        temp . push_back(ConvertIntoString(fi1)) ;
        temp . push_back(ConvertIntoString(fi)) ;
        temp . push_back("E") ;
        table . push_back(temp) ;
        temp . clear() ;
        temp . push_back(ConvertIntoString(fi2)) ;
        temp . push_back(ConvertIntoString(fi)) ;
        temp . push_back("E") ;
        table . push_back(temp) ;
        temp . clear() ;
}

void ReClosure(int &in , int &fi , int in1 , int fi1) {
        in = state + 1 ;
        fi = state + 2 ;
        state += 2 ;
        vector<string> temp ;
        temp . push_back(ConvertIntoString(fi1)) ;
        temp . push_back(ConvertIntoString(in1)) ;
        temp . push_back("E") ;
        table . push_back(temp) ;
        temp . clear() ;
        temp . push_back(ConvertIntoString(in)) ;
        temp . push_back(ConvertIntoString(in1)) ;
        temp . push_back("E") ;
        table . push_back(temp) ;
        temp . clear() ;
```

```
        temp . push_back(ConvertIntoString(in)) ;
        temp . push_back(ConvertIntoString(fi)) ;
        temp . push_back("E") ;
        table . push_back(temp) ;
        temp . clear() ;
        temp . push_back(ConvertIntoString(fi1)) ;
        temp . push_back(ConvertIntoString(fi)) ;
        temp . push_back("E") ;
        table . push_back(temp) ;
        temp . clear() ;
}

void ReSolveRe(int &in , int &fi , string str) {
        str = ConvertIntoPostfix(str) ;
        vector<pair<int , int> > solve ;
        for(int i = 0 ; i < str . length() ; i ++) {
                if(str[i] == 'a' || str[i] == 'b') {
                        Only_a_b(in , fi , str[i]) ;
                        solve . push_back(make_pair(in , fi)) ;
                } else {
                        if(str[i] == '.') {
                                int in2 = solve[solve . size() - 1] . first ;
                                int fi2 = solve[solve . size() - 1] . second ;
                                int in1 = solve[solve . size() - 2] . first ;
                                int fi1 = solve[solve . size() - 2] . second ;
                                solve . pop_back() ;
                                solve . pop_back() ;
                                ReConcatenationRe(in , fi , in1 , fi1 , in2 , fi2) ;
                                solve . push_back(make_pair(in , fi)) ;
                        }
                        if(str[i] == '|') {
                                int in2 = solve[solve . size() - 1] . first ;
                                int fi2 = solve[solve . size() - 1] . second ;
                                int in1 = solve[solve . size() - 2] . first ;
                                int fi1 = solve[solve . size() - 2] . second ;
                                solve . pop_back() ;
                                solve . pop_back() ;
                                ReUnionRe(in , fi , in1 , fi1 , in2 , fi2) ;
                                solve . push_back(make_pair(in , fi)) ;
```

```
                        }
                        if(str[i] == '*') {
                                int in1 = solve[solve . size() - 1] . first ;
                                int fi1 = solve[solve . size() - 1] . second ;
                                solve . pop_back() ;
                                ReClosure(in , fi , in1 , fi1) ;
                                solve . push_back(make_pair(in , fi)) ;
                        }
                }
        }
}

void PrintSol() {
        cout <<"State\ta\tb\tE\n" ;
        for(int i = 0 ; i < table . size() ; i ++) {
                if(table[i][2] == "a") {
                        cout << table[i][0] << "\t" << table[i][1] << "\t_\t_\n" ;
                }
                if(table[i][2] == "b") {
                        cout << table[i][0] << "\t_\t" << table[i][1] << "\t_\n" ;
                }
                if(table[i][2] == "E") {
                        cout << table[i][0] << "\t_\t_\t" << table[i][1] << "\n" ;
                }
        }
}
int main()
{
   string input;
   int in;
   int fi;
   cout<<"********** Enter a Regular Expression **********"<<endl;
   cin>>input;
   ReSolveRe(in,fi,input);
   cout<<endl<<"Intial State ---> "<<in<<endl<<endl<<"Final State  ---> "<<fi<<endl<<endl;
   PrintSol();
   return 0;
}
```

Output :

```
"F:\Semester 6\Labs\Compiler_Design\CD_Program_2.exe"
********** Enter a Regular Expression **********
a|b.a
Postfix Expression for Entered Regular Expression ----> ab|a.

Intial State ---> 4

Final State  ---> 7

State   a       b       E
0       1       _       _
2       _       3       _
4       _       _       0
4       _       _       2
1       _       _       5
3       _       _       5
6       7       _       _
5       _       _       6

Process returned 0 (0x0)   execution time : 5.060 s
Press any key to continue.
```

Program No. 3

Aim :- Perform tasks of lexical analyzer using LEX Tool.

Source Code :-

```
%{
        /*
    Lab Name - Compiler Design
    Objective - Perfrom Lexical Analysis using LEX Tool
    Name - Anup Agrawal
    Roll No. - UE143014
    Date - 15/02/2017
        */
        FILE *out;
        int line = 0;
%}

%%
[\n] {fprintf(out,"  Line Number ----->  %d\n",++line);}
[/][/](.*[ \t]*.*)*[\n] {fprintf(out,"%s,   is a comment. \n\n",yytext);}
[/][*][^*/]*[*][/] {fprintf(out,"%s,   is a multiline comment. \n\n",yytext);}
include|int|float|char|double|long|main {fprintf (out,"%s,   is a keyword.\n\n",yytext);}
[0-9]+ {fprintf(out,"%s,   is a literal. \n\n",yytext);}
[a-zA-Z_][0-9a-zA-Z_]* {fprintf(out,"%s,   is an identifier. \n\n",yytext);}
[0-9!@#$%^&()\*\-\+._=a-zA-Z~|?\\:\[\]{},/"'][0-9a-zA-Z_!@#$%^&()\*\-\+.=~|?\\:\[\]{},/"']+
{fprintf(out,"%s,   is not an identifier. \n\n",yytext);}
[*\-+/=] {fprintf(out,"%s,   is a operator. \n\n",yytext);}
[;] {fprintf(out,"%s,   is a termination symbol. \n\n",yytext);}
[{]+ {fprintf(out,"%s,   is a opening a block. \n\n",yytext);}
[}]+ {fprintf(out,"%s,   is a closing a block. \n\n",yytext);}
%%

main()
{
        yyin = fopen("test.txt","r");
        out = fopen("output.txt","w");
        fprintf(out,"  Line Number ----->  %d\n",++line);
        yylex();
}
```

Contents of file test.txt

```
main
{
        // this is single line comment
        /* this is comment
           comment
           comment
        */
        14anu12p;
        float @@@123anu;
        int a = 10 ;
        int b = 20 ;
        int c = a + b ;
        c = 30;
}
```

Contents of file output.txt

```
  Line Number ----->  1
main,   is a keyword.

  Line Number ----->  2
{,   is a opening a block.

  Line Number ----->  3
// this is single line comment
,   is a comment.

/* this is comment
           comment
           comment
        */,   is a multiline comment.

  Line Number ----->  4
14anu12p,   is not an identifier.

;,   is a termination symbol.

  Line Number ----->  5
```

float, is a keyword.

@@@123anu, is not an identifier.

;, is a termination symbol.

Line Number -----> 6
int, is a keyword.

a, is an identifier.

=, is a operator.

10, is a literal.

;, is a termination symbol.

Line Number -----> 7
int, is a keyword.

b, is an identifier.

=, is a operator.

20, is a literal.

;, is a termination symbol.

Line Number -----> 8
int, is a keyword.

c, is an identifier.

=, is a operator.

a, is an identifier.

+, is a operator.

b, is an identifier.

;, is a termination symbol.

Line Number -----> 9
c, is an identifier.

=, is a operator.

30, is a literal.

;, is a termination symbol.

Line Number -----> 10
}, is a closing a block.

Program No. 4

Aim :- Remove left recursion from a CFG and perform left factoring for a CFG.

Source Code :-

- **Removal of Left Factoring :**

```
/*
  Lab Name - Compiler Design
  Objective - Removal of Left Recursion from a CFG
  Name - Anup Agrawal
  Roll No. - UE143014
  Date - 22/02/2017
*/

#include<bits/stdc++.h>
using namespace std;
vector<string> v;
vector<string> p;
vector<string> a;
vector<string> b;
vector<string> ans;
vector<string> all;

void input()
{
   int n;
   cout<<"*************** Enter Number of Variables
****************************"<<endl;
   cin>>n;
   cout<<"*************** Enter Productions
************************************"<<endl;
   string s;
   for(int i=0;i<n;i++)
   {
      cin>>s;
      v.push_back(s);
   }
}
```

```
vector<string> breakProductions(string t)
{
   vector<string> tempv;
   string temps = "";
   for(int i=2;i<t.length();i++)
   {
      if(t[i] == '|')
      {
         tempv.push_back(temps);
         temps = "";
      }
      else{
         temps = temps + string(1,t[i]);
      }
   }
   tempv.push_back(temps);
   return tempv;
}

void printVector(vector<string> tempv)
{
   for(int i=0;i<tempv.size();i++)
   {
      cout<<tempv[i]<<endl;
   }
   cout<<endl;
}

bool IsLeftRecursive(vector<string> p, char start)
{
   bool flag = false;
   for(int i=0;i<p.size();i++)
   {
      string temps;
      temps = p[i];
      if(temps[0] == start)
      {
         flag = true;
```

```
        }
    }
    if(flag)
    {
        return true;
    }
    else{
        return false;
    }
}

vector<string> AlphaFinder(vector<string> p, char start)
{
    vector<string> tempv;
    for(int i=0;i<p.size();i++)
    {
        string temps;
        temps = p[i];
        if(temps[0] == start)
        {
            temps = temps.substr(1,temps.length()-1);
            tempv.push_back(temps);
        }
    }
    return tempv;
}

vector<string> BetaFinder(vector<string> p, char start)
{
    vector<string> tempv;
    for(int i=0;i<p.size();i++)
    {
        string temps;
        temps = p[i];
        if(temps[0] != start)
        {
            tempv.push_back(temps);
        }
    }
```

```
    return tempv;
}

vector<string> RemoveLeftRecursion(vector<string> a, vector<string> b, char start)
{
    string temps = "";
    string dash = """;
    string sign = "-";
    vector<string> tempv;
    temps = start + sign;
    for(int i=0;i<b.size();i++)
    {
      if(i == (b.size()-1))
        {
            temps = temps + b[i] + start + dash;
        }
      else{
            temps = temps + b[i] + start + dash + '|' ;
        }
    }
    tempv.push_back(temps);
    temps = "";
    temps = start + dash + '-' ;
    for(int i=0;i<a.size();i++)
    {
      if(i == (a.size()-1))
        {
            temps = temps + a[i] + start + dash;
        }
      else{
            temps = temps + a[i] + start + dash + '|' ;
        }
    }
    temps = temps + '|' + '$';
    tempv.push_back(temps);
    return tempv;
}

void leftRecursion()
```

```
{
   for(int i=0;i<v.size();i++)
   {
      string temp;
      char start;
      temp = v[i];
      start = temp[0];
      p = breakProductions(v[i]);
      if(IsLeftRecursive(p,start))
      {
         cout<<"********** Yes, It is a left Recursive Production  "<<v[i]<<"
***********"<<endl;
         a = AlphaFinder(p,start);
         cout<<"******************** Alpha
************************************** "<<endl;
         printVector(a);
         b = BetaFinder(p,start);
         cout<<"******************** Beta
****************************************"<<endl;
         printVector(b);
         ans = RemoveLeftRecursion(a,b,start);
         cout<<"*************** Non-Left Recursive Productions
*********************"<<endl;
         printVector(ans);
         all.insert(all.end(),ans.begin(),ans.end());
      }
      else{
         cout<<"******** No, It is not a Left Recursive Production  "<<v[i]<<"
*************"<<endl;
         all.push_back(v[i]);
      }
   }
   cout<<"**************** Final Grammar **********************"<<endl<<endl;
   printVector(all);
}

int main()
{
   input();
```

```
    leftRecursion();
    cout<<"************ Let's See what happens, in the case of left factoring
********************"<<endl;
    return 0;
}
```

```
"F:\Semester 6\Labs\Compiler_Design\CD_Program_4-1.exe"
*************** Enter Number of Variables ******************************
2
*************** Enter Productions **************************************
E-Eabc|Ab
A-uv|x
********** Yes, It is a left Recursive Production  E-Eabc|Ab ************
********************** Alpha ****************************************
abc

********************** Beta *****************************************
Ab

*************** Non-Left Recursive Productions *********************
E-AbE'
E'-abcE'|$

******** No, It is not a Left Recursive Production  A-uv|x *************
***************** Final Grammar ************************

E-AbE'
E'-abcE'|$
A-uv|x

************ Let's See what happens, in the case of left factoring ********************

Process returned 0 (0x0)   execution time : 20.794 s
Press any key to continue.
```

- **Left Factoring For a CFG :**

```
/*
  Lab Name - Compiler Design
  Objective - Apply Left Factoring for a CFG
  Name - Anup Agrawal
  Roll No. - UE143014
  Date - 22/02/2017
*/

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

```
vector<string> p;
vector<string> a;
vector<string> b;
vector<string> ans;
vector<string> all;

void input()
{
   int n;
   cout<<"*************** Enter Number of Variables
****************************"<<endl;
   cin>>n;
   cout<<"*************** Enter Productions
*************************************"<<endl;
   string s;
   for(int i=0;i<n;i++)
   {
      cin>>s;
      v.push_back(s);
   }
}

vector<string> breakProductions(string t)
{
   vector<string> tempv;
   string temps = "";
   for(int i=2;i<t.length();i++)
   {
      if(t[i] == '|')
      {
         tempv.push_back(temps);
         temps = "";
      }
      else{
         temps = temps + string(1,t[i]);
      }
   }
   tempv.push_back(temps);
   return tempv;
```

```
}

void printVector(vector<string> tempv)
{
   for(int i=0;i<tempv.size();i++)
   {
      cout<<tempv[i]<<endl;
   }
   cout<<endl;
}

vector<string> AlphaFinder(vector<string> p, char start)
{
   int cnt = 0;
   vector<string> tempv;
   string temps = "";
   string temp_s = "";
   bool flag = true;
   int sizee;
   sizee = p.size();
   string temp[sizee];
   for(int i=0;i<p.size();i++)
   {
      temp[i] = p[i];
   }
   while(flag)
   {
     if(temp[0][cnt] == temp[1][cnt])
     {
        flag = true;
        temps = temps + temp[1][cnt];
        cnt++;
     }
     else{
       flag = false;
     }
   }
   for(int i=2;i<p.size();i++)
   {
```

```
        for(int j=0;j<temps.length();j++)
        {
            if(temp[i][j] == temps[j])
            {
                cout<<"aa gya"<<endl;
                temps[j] = temp[i][j];
                cout<<temps<<endl;
            }
            else{
                temps = temps.substr(0,temps.length()-(temps.length()-j));
                cout<<"***********"<<temps<<endl;
                break;
            }
        }
    }
 tempv.push_back(temps);
 return tempv;
}

vector<string> BetaFinder(vector<string> p, char start)
{
  string temps;
  string temp_s;
  vector<string> tempv;
  temps = a[0];
  for(int i=0;i<p.size();i++)
  {
    temp_s = p[i];
    if(temp_s == temps)
    {
        string s = "";
        tempv.push_back(s);
    }
    else{
    temp_s = temp_s.substr(temps.length(),temp_s.length() - temps.length());
    tempv.push_back(temp_s);
    }
  }
  return tempv;
```

```
}

bool IsLeftFactoringPoss(vector<string> p,char start)
{
   int cnt = 0;
   vector<string> tempv;
   string temps = "";
   string temp_s = "";
   bool flag = true;
   int sizee;
   sizee = p.size();
   string temp[sizee];
   for(int i=0;i<p.size();i++)
   {
      temp[i] = p[i];
   }
   while(flag)
   {
     if(temp[0][cnt] == temp[1][cnt])
     {
        flag = true;
        temps = temps + temp[1][cnt];
        cnt++;
     }
     else{
       flag = false;
     }
   }
   for(int i=2;i<p.size();i++)
   {
      for(int j=0;j<temps.length();j++)
      {
         if(temp[i][j] == temps[j])
         {
            temps[j] = temp[i][j];
         }
         else{
            temps = temps.substr(0,temps.length()-(temps.length()-j));
            break;
```

```
            }
        }
    }
    if(temps.size() != 0)
    {
        return true;
    }
    else{
        return false;
    }
}

vector<string> leftFactoredProd(vector<string> a, vector<string> b, char start)
{
    string temps = "";
    string dash = """;
    string sign = "-";
    vector<string> tempv;
    temps = start + sign;
    for(int i=0;i<b.size();i++)
    {
      if(i == (b.size()-1))
        {
            temps =  temps + start + dash + b[i] ;
        }
      else{
            temps = temps + start + dash + b[i] + '|' ;
        }
    }
    tempv.push_back(temps);
    temps = "";
    temps = start + dash + '-' ;
    for(int i=0;i<a.size();i++)
    {
      if(i == (a.size()-1))
        {
            temps = temps + a[i];
        }
      else{
```

```
            temps = temps + a[i] ;
        }
    }
    tempv.push_back(temps);
    return tempv;
}

void leftFactoring()
{
    for(int i=0;i<v.size();i++)
    {
        string temp;
        char start;
        temp = v[i];
        start = temp[0];
        p = breakProductions(v[i]);
        if(IsLeftFactoringPoss(p,start))
        {
        cout<<"******** Yes, Left Factoring can be applied for this  "<<v[i]<<"
*********"<<endl;
        a = AlphaFinder(p,start);
        cout<<"********* Alpha *********"<<endl;
        printVector(a);
        b = BetaFinder(p,start);
        cout<<"******** Beta ********"<<endl;
        printVector(b);
        ans = leftFactoredProd(a,b,start);
        cout<<"********** Productions after apply Left Factoring ***********"<<endl;
        printVector(ans);
        all.insert(all.end(),ans.begin(),ans.end());
        }
        else{
            cout<<"******** No, Left Factoring cannot be applied for this  "<<v[i]<<"
********"<<endl;
            all.push_back(v[i]);
        }
    }
    cout<<"************** Final Grammar *****************"<<endl;
    printVector(all);
```

```
}

int main()
{
   input();
   leftFactoring();
   return 0;
}
```

"F:\Semester 6\Labs\Compiler_Design\CD_Program_4-2.exe"

```
*************** Enter Number of Variables ******************************
2
*************** Enter Productions ****************************************
E-abc|abA
A-ab|cd
******** Yes, Left Factoring can be applied for this  E-abc|abA *********
********* Alpha *********
ab

******** Beta ********
c
A

********** Productions after apply Left Factoring ***********
E-E'c|E'A
E'-ab

******** No, Left Factoring cannot be applied for this  A-ab|cd ********
*************** Final Grammar ******************
E-E'c|E'A
E'-ab
A-ab|cd


Process returned 0 (0x0)   execution time : 15.008 s
Press any key to continue.
```

Program No. 5

Aim :- Implement LL(1) Top - Down Parser for a LL(1) grammar.

Source Code :-

```
/*
  Lab Name - Compiler Design
  Objective - LL(1) Parser
  Name - Anup Agrawal
  Roll No. - UE143014
  Date - 22/02/2017
*/

#include<bits/stdc++.h>
using namespace std;
void line(int ter)
{
   ter=ter*5;
   cout<<"  ";
   for(int i=1; i<=ter; i++)
      cout<<"--";
   cout<<"\n";
}

void displayStack(stack<char> a)
{
   stack <char> b;
   string s,temp;
   while(!a.empty())
   {
      temp=a.top();
      s.insert(0,temp);
      b.push(a.top());
      a.pop();
   }
   cout<<s;
   while(!b.empty())
   {
      a.push(b.top());
```

```
        b.pop();
    }
}
void displayQueue(queue<char> a)
{
    queue <char> b;
    string s,temp;

    while(!a.empty())
    {
        temp=a.front();
        s+=temp;
        b.push(a.front());
        a.pop();
    }
    cout<<s;
    while(!b.empty())
    {
        a.push(b.front());
        b.pop();
    }

}
int main()
{
    int i=0,n=0,j,k,flag=1,update=0,epsilon=1;
    char ch,ch1;
    string temp,temp1;
    string str[10],tok[20][20],var[10],tok1[20][20],c,parse[50],table[20][20];
    fstream fin,fout;
    fin.open("output3.txt",ios::in);
    unordered_map <string, string> m;
    unordered_map <string, string> q;
    unordered_map <string,int> row;
    unordered_map <string,int> column;
    string t[20][20];
    string first[10][2];
    string terminal[20];
    int ter=0;
```

```
    while(!fin.eof())
    {
        getline(fin,str[i]);
        if(!str[i].empty())
        {
            n++;
            var[i]=str[i][0];
            first[i][0]=var[i];
            i++;
        }
    }


cout<<"*********************************************************************"
<<endl;
    for(i=0; i<n; i++)
    {
        k=0;
        for(j=2; str[i][j]!='\0'; j++)
        {
            if(str[i][j]=='|'||str[i][j]=='>')
            {
                j++;
                for(; str[i][j]!='\0'&&str[i][j]!='|'; j++)
                {
                    tok[i][k]+=str[i][j];
                    tok1[i][k]+=str[i][j];
                }
                j--;
                k++;
            }
        }
    }
//first code-------------------------------------------------------
    do {
        update=0;
        for(i=0; i<n; i++)
        {
```

```
for(j=0; (!(tok[i][j].empty())); j++)
{
   ch=tok[i][j][0];
   if(!(isupper(ch))&&m[var[i]].find(ch)==-1)
   {
      m[var[i]]+=ch;
      t[i][j]+=ch;
      update=1;
   }
   if(isupper(ch))
   {
      k=0;
      c;
      c+=ch;
      while(m[c][k]!='\0')
      {
         if(m[var[i]].find(m[c][k])==-1&&m[c][k] != 'e')
         {
            m[var[i]]+=m[c][k];
            t[i][j]+=m[c][k];

            update=1;

         }
         else if(m[c][k]=='e')
         {
            tok[i][j].erase(tok[i][j].begin());
            if(tok[i][j].empty())
            {
               m[var[i]]+='e';
               t[i][j]+=m[var[i]];

            }
            else
               j--;
         }
         k++;
      }
      c.clear();
```

```
            }
        }
    }

} while(update==1);
//first code ends---------------------------------------------
cout<<"=======<<"First:"<<"=======<<endl;
for(i=0; i<n; i++)
{
    first[i][1]+=m[var[i]];
    cout<<var[i]<<":"<<first[i][1]<<endl;
}

//follow code starts---------------------------------------------
q[var[0]]+='$';

do {
    update=0;
    for(i=0; i<n; i++)
    {
        for(j=0; (!(tok1[i][j].empty())); j++)
        {
            for(k=0; tok1[i][j][k]!='\0'; k++)
            {
                string c1;
                ch=tok1[i][j][k];

                int next=k;
                epsilon=1;
                if(isupper(ch))
                {
                    while(epsilon==1)
                    {
                        epsilon=0;

                        next=next+1;
                        c+=ch;
                        ch1=tok1[i][j][next];
                        // cout<<ch1<<endl;
```

```
c1+=ch1;
if(isupper(ch1))
{
   int l=0;

   while(m[c1][l]!='\0')
   {
      if(q[c].find(m[c1][l])==-1&&m[c1][l]!='e')
      {
         q[c]+=m[c1][l];
         update=1;
      }
      else if(m[c1][l]=='e')
      {
         epsilon=1;
      }

      l++;
   }


}
else if(q[c].find(ch1)==-1&&ch1!='e')
{
   q[c]+=ch1;
   update=1;
}
else if(ch1=='\0')
{
   int l=0;

   while(q[var[i]][l]!='\0')
   {
      if(q[c].find(q[var[i]][l])==-1)
      {
         q[c]+=q[var[i]];
         update=1;

      }
```

```
                    l++;

                }
            }

            c.clear();
            c1.clear();
        }
    }
  }
 }
}

} while(update==1);

cout<<"=======<<"Follow:"<<"=======<<endl;
for(i=0; i<n; i++)
{

   cout<<var[i]<<":"<<q[var[i]]<<endl;
}

//Follow code ends-----------------------------------------------------

//Parse table starts-----------------------------------------------

for(i=0; i<n; i++)
   for(j=0; (!(tok1[i][j].empty())); j++)
      for(k=0; tok1[i][j][k]!='\0'; k++)
      {
         ch=tok1[i][j][k];
         if(!isupper(ch)&&ch!='e'&&temp1.find(ch)==-1)
         {
            terminal[ter]=ch;
            temp1+=ch;
            ter++;
         }
      }
terminal[ter++]='$';
```

```
for(i=0; i<n; i++)
{
   row[var[i]]=i;
}
for(i=0; i<ter; i++)
{
   column[terminal[i]]=i;
}

for(i=0; i<n; i++)
{
   for(j=0; j<ter; j++)
   {
      for(k=0; (!(tok1[i][k].empty())); k++)
      {
         for(int z=0; t[i][k][z]!='\0'; z++)
         {
            string h;
            h=t[i][k][z];
            if(h==terminal[j])
            {
               int t1;

               t1=column[h];
               table[i][t1]=tok1[i][k];
            }
         }
      }
   }
}
for(i=0; i<n; i++)
{
   if(m[var[i]].find('e')==1)
   {
      string t2;
      int len;
      t2=q[var[i]];
      len=t2.length();
      for(j=0; j<len; j++)
```

```
            {
                string t3;
                t3=t2[j];
                char t4=t2[j];

                if(!(t4=='\0'))
                    table[i][column[t3]]='e';

            }
        }
    }
    for(i=0; i<n; i++)
    {
        for(j=0; j<ter; j++)
        {
            if(!(table[i][j].empty()))
            {
                string str;
                str+=var[i];
                str+="->";
                table[i][j].insert(0,str);
            }

        }
    }
    cout<<endl<<"================================Parse
Table================================="<<"\n\n\n";
    for(i=0; i<ter; i++)
        cout<<setw(10)<<terminal[i];
    cout<<endl;
    line(ter);
    for(i=0; i<n; i++)
    {
        cout<<var[i];
        for(j=0; j<ter; j++)
        {
            cout<<setw(10)<<table[i][j];
        }
        cout<<endl;
```

```
        line(ter);
    }
    cout<<"\n\n\n";
//-----------------------------------------Parse table ends-----------------------------------------
    string pstring,terminal1;
    for(i=0; i<ter; i++)
    {
        terminal1+=terminal[i];
    }

    char s1,ans;
    int a1=0,error=0;
    s1=var[0][0];
    stack<char> stk;
    queue<char> que;
    stk.push('$');
    stk.push(s1);
    char qtop,stop;


    do {
        cout<<"Enter the string to parse:";
        cin>>pstring;
        string st,qt,tval;

        while(pstring[a1]!='\0')
        {
            que.push(pstring[a1]);
            a1++;
        }
        que.push('$');
        while(!que.empty()&&!stk.empty())
        {
            displayStack(stk);
            cout<<"\t\t\t";
            displayQueue(que);
            cout<<"\t\t\t";
            cout<<tval;
            tval.erase();
```

```
cout<<endl;
qtop=que.front();

stop=stk.top();
if(qtop==stop)
{
   que.pop();
   stk.pop();
}
else if(stop=='e')
{
   stk.pop();
}
else if(qtop!=stop)
{

   if(terminal1.find(stop)==1)
   {
      error=1;
      break;
   }
   else
   {
      st=stop;
      qt=qtop;
      int r1,c1;
      r1=row[st];
      c1=column[qt];
      tval=table[r1][c1];
      if(tval.empty())
      {
         error=1;
         cout<<"Error";
         break;
      }
      else
      {
         stk.pop();
         i=0;
```

```
                while(tval[i]!='\0')
                    i++;
                i--;
                while(tval[i]!='>')
                {
                    if(tval[i]!=' ')
                    {
                        stk.push(tval[i]);
                    }
                    i--;
                }
            }

        }
    }

    }
    if(error==0&&que.empty()&&stk.empty())
        cout<<"String is parsed";
    else
    {
        cout<<"String not parsed";
    }
    cout<<"\nDo you want to parse more string(y/n):";
    cin>>ans;
  } while(ans=='Y'||ans=='y');
}
```

Input Grammar :

S->ABCD
A->a|e
B->b|e
C->c|e
D->d

Output :

```
"F:\Semester 6\Labs\Compiler_Design\CD_Program_5.exe"
*************************************************************************
=======First:=======
S:abcd
A:ae
B:be
C:ce
D:d
=======Follow:=======
S:$
A:bcd
B:cd
C:d
D: $

================================Parse Table================================


        a         b         c         d         $
  ------------------------------------------------
S   S->ABCD   S->ABCD   S->ABCD   S->ABCD
  ------------------------------------------------
A      A->a      A->e      A->e      A->e
  ------------------------------------------------
B                B->b      B->e      B->e
  ------------------------------------------------
C                          C->c      C->e
  ------------------------------------------------
D                                    D->d
  ------------------------------------------------
```

```
Enter the string to parse:abcd
$S                          abcd$
$DCBA                       abcd$                    S->ABCD
$DCBa                       abcd$                    A->a
$DCB                        bcd$
$DCb                        bcd$                     B->b
$DC                         cd$
$Dc                         cd$                      C->c
$D                          d$
$d                          d$                       D->d
$                           $
String is parsed
Do you want to parse more string(y/n):
```

Program No. 6

Aim :- Implement Top-Down Recursive Descendant Parser.

Source Code :-

```
/*
  Lab Name - Compiler Design
  Objective -  Descendant Descent
  Name - Anup Agrawal
  Roll No. - UE143014
  Date - 01/03/2017
*/
#
include < bits / stdc++.h >
 using namespace std;

int F(string str, int i) {
 if (str[i] == 'a') {
  if (i == str.length() - 1) {
   return 1;
  } else {
   if (F(str, i + 1) == 1)
    return 1;
   else
    return 0;
  }
 } else {
  return 0;
 }
}

int T(string str, int i) {
 if (str[i] == 'b') {
  if (i == str.length() - 1) {
   return 1;
  } else {
   if (T(str, i + 1) == 1)
    return 1;
   else
```

```
      return 0;
    }
  } else {
    return 0;
  }
}

void E(string str) {
  if (str[0] == '+') {
    int val = F(str, 1);
    if (val == 1)
      cout << "Valid string ...\n";
    else {
      val = T(str, 1);
      if (val == 1)
        cout << "Valid string ...\n";
      else
        cout << "Invalid string ...\n";
    }
  } else {
    cout << "Invalid string ...\n";
  }
}

int main() {
  /*
    E->+F|+T
    F->aF|$
    T->bT|$
  */
  cout << "Enter any string to parse -\n";
  string str;
  cin >> str;
  E(str);
  return 0;
}
```

Output :-

```
"F:\Semester 6\compiler design\Lab\Compiler_Design\CD_Program_6.exe"
Enter any string to parse -
++a
Invalid string ...

Process returned 0 (0x0)   execution time : 4.949 s
Press any key to continue.
```

```
"F:\Semester 6\compiler design\Lab\Compiler_Design\CD_Program_6.exe"
Enter any string to parse -
+a
Valid string ...

Process returned 0 (0x0)   execution time : 4.984 s
Press any key to continue.
```

Program No. 7

Aim :- Implement the idea of Bottom-Up Parser.

Source Code :-

```
/*
  Lab Name - Compiler Design
  Objective - Bottom-Up parser
  Name - Anup Agrawal
  Roll No. - UE143014
  Date - 05/04/2017
*/


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

vector < string > v;
string instr;
stack < char > stk;
queue < char > que;
string action;
string rule;
int error = 0;
char qtop;
char stop;
char ch = 'Y';

void shift();
void reduce();

void displayStack(stack < char > a) {
 stack < char > b;
 string s, temp;
 while (!a.empty()) {
   temp = a.top();
   s.insert(0, temp);
   b.push(a.top());
   a.pop();
```

```
  }
  cout << s;
  while (!b.empty()) {
    a.push(b.top());
    b.pop();
  }
}

void displayQueue(queue < char > a) {
  queue < char > b;
  string s, temp;

  while (!a.empty()) {
    temp = a.front();
    s += temp;
    b.push(a.front());
    a.pop();
  }
  cout << s;
  while (!b.empty()) {
    a.push(b.front());
    b.pop();
  }
}

void input() {
  string str;
  int n;
  cout << "*************** Enter Number of Variables ******************************"
<< endl;
  cin >> n;
  cout << "*************** Enter Productions ************************************"
<< endl;
  for (int i = 0; i < n; i++) {
    cin >> str;
    v.push_back(str);
  }
  cout << "Enter a input string" << endl;
  cin >> instr;
```

```
}

void reverseStr(string & a) {
  int n = a.length();
  for (int i = 0; i < n / 2; i++)
    swap(a[i], a[n - i - 1]);
}

void display(string action, string rule) {
  displayStack(stk);
  cout << "\t\t\t\t";
  displayQueue(que);
  cout << "\t\t\t\t";
  cout << action;
  cout << "\t\t\t\t";
  cout << rule << endl << endl;
}

void shift() {
  if (que.front() != '$') {
    qtop = que.front();
    stk.push(qtop);
    que.pop();
    display("shift", "None");
    reduce();
  } else if (stk.top() == 'S' && stk.size() == 2 && que.front() == '$') {
    cout << "String is Successfully Parsed" << endl;
  } else {
    cout << "String is NOT Parsed" << endl;
  }
}

void reduce() {
  int update = 0;
  stack < char > b;
  string s, temp;
  bool flag = false;
  if (stk.top() != '$' && stk.top() != 'S') {
    do {
```

```
  update = 0;
  flag = false;
  for (int i = 1; i < stk.size(); i++) {
    for (int j = 1; j <= i; j++) {
      update = 0;
      flag = false;
      temp = stk.top();
      s.insert(0, temp);
      b.push(stk.top());
      stk.pop();
    }
    for (int k = 0; k < v.size(); k++) {
      string temps;
      string tempss;
      char inc;
      temps = v[k];
      tempss = temps;
      temps = temps.substr(2, temps.length() - 2);
      if (temps == s) {
        flag = true;
        s = "";
        update = 1;
        inc = tempss[0];
        stk.push(inc);
        display("reduce", tempss);
        while (!b.empty()) {
          b.pop();
        }
      }
    }
    if (flag == false) {
      s = "";
      while (!b.empty()) {
        stk.push(b.top());
        b.pop();
      }
    }
  }
} while (update == 1);
```

```
    shift();
  } else if (stk.top() == 'S') {
    cout << "String is Successfully Parsed" << endl;
  } else {
    cout << "String is NOT Parsed" << endl;
  }
}

void intialize() {
  cout << "****************** BUP ******************" << endl;
  input();
  int i = 0;
  while (instr[i] != '\0') {
    que.push(instr[i]);
    i++;
  }
  que.push('$');
  stk.push('$');
  cout << "Stack" << "\t\t\t\t" << "Queue" << "\t\t\t\t" << "Action" << "\t\t\t\t" << "Production"
<< endl << endl;
  display("None", "None");
}

void allclear() {
  v.clear();
  while (!stk.empty()) {
    stk.pop();
  }
  while (!que.empty()) {
    que.pop();
  }
}

int main() {
  while (ch == 'Y' || ch == 'y') {
    intialize();
    shift();
    cout << "Wnat to parse more strings, please enter Y/y" << endl;
    cin >> ch;
```

```
    if (ch == 'y' || ch == 'Y') {
      system("cls");
      allclear();
    } else {
      exit(0);
    }
  }
  return 0;
}
```

Output :-

```
"F:\Semester 6\compiler design\Lab\Compiler_Design\CD_Program_7.exe"
******************* BUP *******************
*************** Enter Number of Variables ******************************
4
*************** Enter Productions ****************************************
S-T
T-F+U+F
U-F*F
F-i
Enter a input string
i+i*i+i
Stack                          Queue                          Action                         Production

$                              i+i*i+i$                               None                           None

$i                             +i*i+i$                        shift                          None

$F                             +i*i+i$                        reduce                         F-i

$F+                            i*i+i$                         shift                          None

$F+i                           *i+i$                          shift                          None

$F+F                           *i+i$                          reduce                         F-i

$F+F*                          i+i$                           shift                          None

$F+F*i                         +i$                            shift                          None

$F+F*F                         +i$                            reduce                         F-i
```

```
$F+F                    *i+i$                   reduce                  F-i

$F+F*                   i+i$                    shift                   None

$F+F*i                  +i$                     shift                   None

$F+F*F                  +i$                     reduce                  F-i

$F+U                    +i$                     reduce                  U-F*F

$F+U+                   i$                      shift                   None

$F+U+i                  $                       shift                   None

$F+U+F                  $                       reduce                  F-i

$T                      $                       reduce                  T-F+U+F

$S                      $                       reduce                  S-T

String is Successfully Parsed
Wnat to parse more strings, please enter Y/y
```

Program No. 8

Aim :- Implement Bottom-Up static Operator Precedence Parser.

Source Code :-

```
/*
   Lab Name - Compiler Design
   Objective - Static Operator Precedence Parser
   Name - Anup Agrawal
   Roll No. - UE143014
   Date - 19/04/2017
*/

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

stack < int > resultstk;
string tab[5][5];
stack < char > stk;
stack < char > out;
queue < char > que;
char ch = 'Y';
char qtop;
char stop;
int operation = 0;
string infix = "";
string postfix;
string instr;
stack < string > poststk;

void display(string);
void reduce();
void shift();

void line(int ter) {
  ter = ter * 5;
  cout << "  ";
  for (int i = 1; i <= ter; i++)
    cout << "--";
```

```
  cout << "\n";
}

void displayStack(stack < char > a) {
  stack < char > b;
  string s, temp;
  while (!a.empty()) {
    temp = a.top();
    s.insert(0, temp);
    b.push(a.top());
    a.pop();
  }
  cout << s;
  while (!b.empty()) {
    a.push(b.top());
    b.pop();
  }
}

void displayQueue(queue < char > a) {
  queue < char > b;
  string s, temp;

  while (!a.empty()) {
    temp = a.front();
    s += temp;
    b.push(a.front());
    a.pop();
  }
  cout << s;
  while (!b.empty()) {
    a.push(b.front());
    b.pop();
  }
}

void shift() {
  if (que.front() != '$') {
    qtop = que.front();
```

```
    stk.push(qtop);
    que.pop();
    display("shift");
  } else if (stk.top() == '$' && que.front() == '$') {
    cout << "String is Successfully Parsed" << endl;
  } else {
    cout << "String is NOT Parsed" << endl;
  }
}

bool Isoperator(char ch) {
  if (ch == '+') {
    operation = 1;
    return true;
  } else if (ch == '-') {
    operation = 2;
    return true;
  } else if (ch == '*') {
    operation = 3;
    return true;
  } else if (ch == '/') {
    operation = 4;
    return true;
  } else {
    return false;
  }
}

bool Isoperand(char ch) {
  if (ch == '1' || ch == '2' || ch == '3' || ch == '4') {
    return true;
  } else {
    return false;
  }
}

void reduce() {
  if (stk.top() != '$') {
    out.push(stk.top());
```

```
    stk.pop();
    display("Reduce");
  } else if (stk.top() == 'S') {
    cout << "String is Successfully Parsed" << endl;
  } else {
    cout << "String is NOT Parsed" << endl;
  }
}

void input() {
  string str;
  cout << "Enter a input string" << endl;
  cin >> instr;
}

void table() {
  cout << "Enter in table function" << endl;
  tab[0][0] = "#";
  tab[0][1] = "i";
  tab[0][2] = "+";
  tab[0][3] = "*";
  tab[0][4] = "$";

  tab[1][0] = "i";
  tab[2][0] = "+";
  tab[3][0] = "*";
  tab[4][0] = "$";

  tab[1][1] = "";
  tab[1][2] = ">";
  tab[1][3] = ">";
  tab[1][4] = ">";

  tab[2][1] = "<";
  tab[2][2] = ">";
  tab[2][3] = "<";
  tab[2][4] = ">";

  tab[3][1] = "<";
```

```
  tab[3][2] = ">";
  tab[3][3] = ">";
  tab[3][4] = ">";

  tab[4][1] = "<";
  tab[4][2] = "<";
  tab[4][3] = "<";
  tab[4][4] = "";

  for (int i = 0; i < 5; i++) {
    for (int j = 0; j < 5; j++) {
      cout << tab[i][j] << "\t";
    }
    cout << endl;
  }
}

void display(string action) {
  displayStack(stk);
  cout << "\t\t\t";
  displayQueue(que);
  cout << "\t\t\t";
  cout << action;
  cout << "\t\t\t";
  displayStack(out);
  cout << endl;
}

int findindicesrow(string d) {
  int c = 0;
  for (int i = 0; i < 5; i++) {
    if (tab[0][i] == d) {
      c = i;
    }
  }
  return c;
}

int findindicescolumn(string d) {
```

```
  int c = 0;
  for (int i = 0; i < 5; i++) {
    if (tab[i][0] == d) {
      c = i;
    }
  }
  return c;
}

void PostToIn() {
  string temp;
  while (!out.empty()) {
    temp = out.top();
    postfix.insert(0, temp);
    out.pop();
  }
  char ch;
  int temp1;
  int first;
  int second;
  int result;
  string op1;
  string op2;
  for (int i = 0; i < postfix.length(); i++) {
    ch = postfix[i];
    if (Isoperand(ch)) {
      temp = ch;
      stringstream convertch(temp);
      convertch >> temp1;
      poststk.push(temp);
      resultstk.push(temp1);
    } else if (Isoperator(ch)) {
      op1 = poststk.top();
      first = resultstk.top();
      poststk.pop();
      resultstk.pop();
      op2 = poststk.top();
      second = resultstk.top();
      poststk.pop();
```

```
      resultstk.pop();
      infix = op2 + ch + op1;
      poststk.push(infix);
      switch (operation) {
      case 1:
        resultstk.push(second + first);
        cout << "\t\t\t\t\t\t\t\t\t" << infix << "," << second << "+" << first << "---->" <<
resultstk.top() << endl;
        break;
      case 2:
        resultstk.push(second - first);
        cout << "\t\t\t\t\t\t\t\t\t" << infix << "," << second << "-" << first << "---->" <<
resultstk.top() << endl;
        break;
      case 3:
        resultstk.push(second * first);
        cout << "\t\t\t\t\t\t\t\t\t" << infix << "," << second << "*" << first << "---->" <<
resultstk.top() << endl;
        break;
      case 4:
        resultstk.push(second / first);
        cout << "\t\t\t\t\t\t\t\t\t" << infix << "," << second << "/" << first << "---->" <<
resultstk.top() << endl;
        break;
      }
    }
  }
  for (int i = 0; i < infix.length(); i++) {
    out.push(infix[i]);
  }
  display("None");
  cout << "\t\t\t\t\t\t\t\t\t" << resultstk.top() << " --Answer" << endl;
}

void traverse() {
  cout << "In traverse " << endl;
  string tempqtop;
  string tempstop;
  int row;
```

```
  int column;
  for (int i = 0; i < 2 * instr.length(); i++) {
    qtop = que.front();
    if (qtop == '1' || qtop == '2' || qtop == '3' || qtop == '4') {
      qtop = 'i';
    }
    tempqtop = qtop;
    stop = stk.top();
    if (stop == '1' || stop == '2' || stop == '3' || stop == '4') {
      stop = 'i';
    }
    tempstop = stop;
    column = findindicesrow(tempqtop);
    row = findindicescolumn(tempstop);
    if (tab[row][column] == "<") {
      shift();
    } else if (tab[row][column] == ">") {
      reduce();
    } else {
      cout << "************** ERROR *************" << endl;
    }
  }
}

void intialize() {
  cout << "******************* BUP *******************" << endl;
  input();
  int i = 0;
  while (instr[i] != '\0') {
    que.push(instr[i]);
    i++;
  }
  que.push('$');
  stk.push('$');
  cout << "Stack" << "\t\t\t" << "Queue" << "\t\t\t" << "Action\t\t\t" << "Output" << endl <<
endl;
  display("None");
}
```

```
void allclear() {
  while (!stk.empty()) {
    stk.pop();
  }
  while (!que.empty()) {
    que.pop();
  }
  while (!out.empty()) {
    out.pop();
  }
  while (!resultstk.empty()) {
    out.pop();
  }
  while (!poststk.empty()) {
    out.pop();
  }
  infix = "";
  postfix = "";
  operation = 0;
}

int main() {
  while (ch == 'Y' || ch == 'y') {
    intialize();
    table();
    traverse();
    PostToIn();
    cout << "Wnat to parse more strings, please enter Y/y" << endl;
    cin >> ch;
    if (ch == 'y' || ch == 'Y') {
      system("cls");
      allclear();
    } else {
      exit(0);
    }
  }
  return 0;
}
```

Output :-

```
"F:\Semester 6\compiler design\Lab\Compiler_Design\CD_Program_8.exe"
******************* BUP *******************
Enter a input string
1+4*3+1
Stack                    Queue                    Action                    Output

$                        1+4*3+1$                         None
Enter in table function
#       i       +        *        $
i               >        >        >
+       <       >        <        >
*       <       >        >        >
$       <       <        <
In traverse
$1                       +4*3+1$                  shift
$                        +4*3+1$                  Reduce                    1
$+                       4*3+1$                   shift                     1
$+4                      *3+1$                    shift                     1
$+                       *3+1$                    Reduce                    14
$+*                      3+1$                     shift                     14
$+*3                     +1$                      shift                     14
$+*                      +1$                      Reduce                    143
$+                       +1$                      Reduce                    143*
$                        +1$                      Reduce                    143*+
$+                       1$                       shift                     143*+
$+1                      $                        shift                     143*+
$+                       $                        Reduce                    143*+1
$                        $                        Reduce                    143*+1+
```

```
In traverse
$1                       +4*3+1$                  shift
$                        +4*3+1$                  Reduce                    1
$+                       4*3+1$                   shift                     1
$+4                      *3+1$                    shift                     1
$+                       *3+1$                    Reduce                    14
$+*                      3+1$                     shift                     14
$+*3                     +1$                      shift                     14
$+*                      +1$                      Reduce                    143
$+                       +1$                      Reduce                    143*
$                        +1$                      Reduce                    143*+
$+                       1$                       shift                     143*+
$+1                      $                        shift                     143*+
$+                       $                        Reduce                    143*+1
$                        $                        Reduce                    143*+1+
                                                                            4*3,4*3---->12
                                                                            1+4*3,1+12---->13
                                                                            1+4*3+1,13+1---->14
$                        $                        None                      1+4*3+1
                                                                            14 --Answer
Wnat to parse more strings, please enter Y/y
```

Program No. 9

Aim :- Implement Bottom-Up dynamic Operator Precedence Parser.

Source Code :-

```
/*
  Lab Name - Compiler Design
  Objective - Dynamic Operator Precedence Parser
  Name - Anup Agrawal
  Roll No. - UE143014
  Date - 19/04/2017
*/

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

stack<int> resultstk;
stack<char> stk;
stack<char> out;
queue<char> que;
char qtop;
char stop;
int operation = 0;
string infix = "";
string postfix;
string instr;
stack<string> poststk;
void display(string);
void reduce();
void shift();
char ch ='Y';
set<char> ter;
vector<char> terminal;
vector<string> rightp;
vector<string> v;
vector<char> leftp;
map<char, vector<string> > prod;
map<char , vector<char> > first ;
map<char , vector<char> > last ;
```

```
map<char,int> row;
map<char,int> index;
map<char,int> column;
string tab[20][20];

void displayStack(stack<char> a)
{
   stack <char> b;
   string s,temp;
   while(!a.empty())
  {
      temp=a.top();
      s.insert(0,temp);
      b.push(a.top());
      a.pop();
  }
  cout<<s;
  while(!b.empty())
  {
      a.push(b.top());
      b.pop();
  }
}

void displayQueue(queue<char> a)
{
   queue <char> b;
  string s,temp;

  while(!a.empty())
  {
      temp=a.front();
      s+=temp;
      b.push(a.front());
      a.pop();
  }
  cout<<s;
  while(!b.empty())
  {
```

```
      a.push(b.front());
      b.pop();
   }
}

bool Isoperator(char ch)
{
       if(ch == '+')
       {
          operation = 1;
          return true;
       }
       else if(ch == '-')
       {
          operation = 2;
          return true;
       }
       else if(ch == '*')
       {
          operation = 3;
          return true;
       }
       else if(ch == '/')
       {
          operation = 4;
          return true;
       }
    else{
       return false;
    }
}

bool Isoperand(char ch)
{
    if(ch =='1' || ch =='2' ||  ch =='3' ||  ch =='4')
    {
       return true;
    }
    else{
```

```
        return false;
    }
}

void firstFinder(char ch) {
        vector<string> production = prod[ch] ;
        set<char> ans ;
        for(int i=0 ; i<production.size() ; i++) {
                int j = 0 ;
                string p_str = production[i] ;
                while(j < p_str.length()) {
                        int asc = (int)p_str[j] ;
                        if(asc < 65 || asc > 90) {
                                ans.insert(p_str[j]) ;
                                ter.insert(p_str[j]);
                                break ;
                        } else {
                                if(p_str[j] != ch) {
                                        firstFinder(p_str[j]) ;
                                        vector<char> temp = first[p_str[j]] ;
                                        for(int k=0 ; k<temp.size() ; k ++){
                                                ans.insert(temp[k]);
                                                ter.insert(temp[k]);
                                        }
                                }
                        }
                        j ++ ;
                }
        }
        vector<char> temp ;
        for(set<char> :: iterator it = ans . begin() ; it != ans . end() ; it ++)
    {
                temp.push_back(*it) ;
    }
        first[ch] = temp ;
}

void allclear()
{
```

```
    v.clear();
    while(!stk.empty())
    {
        stk.pop();
    }
    while(!que.empty())
    {
        que.pop();
    }
    while(!out.empty())
    {
        out.pop();
    }
    while(!resultstk.empty())
    {
        out.pop();
    }
    while(!poststk.empty())
    {
        out.pop();
    }
    infix = "";
    postfix = "";
    operation =0;
}

void lastFinder(char ch) {
        vector<string> production = prod[ch] ;
        set<char> ans ;
        for(int i = 0 ; i < production.size() ; i ++) {
                string p_str = production[i] ;
                int j = p_str.length()-1 ;
                while(j >= 0) {
                        int asc = (int)p_str[j] ;
                        if(asc < 65 || asc > 90) {
                                ans.insert(p_str[j]) ;
                                ter.insert(p_str[j]) ;
                                break ;
                        } else {
```

```
                    if(p_str[j] != ch) {
                        lastFinder(p_str[j]) ;
                        vector<char> temp = last[p_str[j]] ;
                        for(int k = 0 ; k < temp . size() ; k ++){
                            ans.insert(temp[k]) ;
                            ter.insert(temp[k]) ;
                    }
                }
                j -- ;
            }
        }
        vector<char> temp ;
        for(set<char> :: iterator it = ans . begin() ; it != ans . end() ; it ++)
    {
        temp.push_back(*it) ;
    }
        last[ch] = temp ;
}
}

void shift()
{
    if(que.front() != '$')
    {
        qtop = que.front();
        stk.push(qtop);
        que.pop();
        display("shift");
    }
    else if(stk.top() == '$' && que.front() == '$')
{
    cout<<"String is Successfully Parsed"<<endl;
}

    else{
     cout<<"String is NOT Parsed"<<endl;
    }
}
```

```
void reduce()
{
   if(stk.top() != '$')
{
   out.push(stk.top());
   stk.pop();
   display("Reduce");
}
else if(stk.top() == 'S')
{
   cout<<"String is Successfully Parsed"<<endl;
}

else{
    cout<<"String is NOT Parsed"<<endl;
}
}

void table()
{
   cout<<"*******Parse table is ********************"<<endl;
   for(int i=0;i<terminal.size();i++)
   {
      row[terminal[i]]=i+1;
   }
   for(int i=0;i<terminal.size();i++)
   {
      column[terminal[i]]=i+1;
   }
   for(int i=1;i<=terminal.size();i++)
   {
      for(int j=1;j<=terminal.size();j++)
      {
         tab[i][j] = "-";
      }
   }
   tab[0][0] = "#";
   string ha= "";
```

```
for(int i=1;i<=terminal.size();i++)
{
   ha = ha + terminal[i-1];
   tab[i][0] = ha;
   ha = "";
}
for(int i=1;i<=terminal.size();i++)
{
   ha = ha + terminal[i-1];
   tab[0][i] = terminal[i-1];
   ha = "";
}

for(int i=0;i<leftp.size();i++)
{
   char tempc = leftp[i];
   string temps = rightp[i];
   int length = temps.length();
   for(int j=0;j<length - 1;j++)
   {
      int asc1 = (int)temps[j] ;
      int asc2 = (int)temps[j + 1] ;
      int asc3 = (int)temps[j + 2] ;
      if((asc1 < 65 || asc1 > 90) && (asc2 < 65 || asc2 > 90))
      {
      int i1 = row[temps[j]] ;
      int i2 = column[temps[j + 1]] ;
      tab[i1][i2] = "=" ;
      }
      if((j < length - 2) && (asc1 < 65 || asc1 > 90) && (asc2 >= 65 && asc2 <= 90) && (asc3
< 65 || asc3 > 90)) {
      int i1 = row[temps[j]] ;
      int i2 = column[temps[j + 2]] ;
      tab[i1][i2] = "=" ;
      }
      if((asc1 < 65 || asc1 > 90) && (asc2 >= 65 && asc2 <= 90))
      {
         vector<char> firsts = first[temps[j + 1]] ;
         for(int k = 0 ; k < firsts . size() ; k ++)
```

```
            {
            int i1 = row[temps[j]] ;
            int i2 = column[firsts[k]] ;
            tab[i1][i2] = "<" ;
            }
      }
      if((asc1 >= 65 && asc1 <= 90) && (asc2 < 65 || asc2 > 90))
      {
         vector<char> lasts = last[temps[j]] ;
         for(int k = 0 ; k < lasts . size() ; k ++)
          {
            int i1 = row[lasts[k]] ;
            int i2 = column[temps[j + 1]] ;
            tab[i1][i2] = ">" ;
         }
      }
}
}
vector<char> firsts = first[v[0][0]] ;
for(int j = 0 ; j < firsts . size() ; j ++)
  {
      int i1 = row['$'] ;
      int i2 = column[firsts[j]] ;
      tab[i1][i2] = "<" ;
  }
vector<char> lasts = last[v[0][0]] ;
for(int j = 0 ; j < lasts . size() ; j ++)
{
  int i1 = row[lasts[j]] ;
  int i2 = column['$'] ;
  tab[i1][i2] = ">" ;
}
cout<<endl<<endl;
for(int i=0;i<=terminal.size();i++)
{
    for(int j=0;j<=terminal.size();j++)
    {
       cout<<tab[i][j]<<"\t";
    }
```

```
        cout<<endl;
    }
}


vector<string> divideProduction(string str)
{
   vector<string> parts ;
   string temp = "" ;
   for(int i = 2;i<str.length();i ++) {
     if(str[i] == '|') {
        parts.push_back(temp) ;
        temp = "" ;
     } else {
         temp = temp + str[i] ;
     }
  }
   if(temp != "") {
    parts.push_back(temp) ;
  }
   return parts ;
}

void display(string action)
{
          displayStack(stk);
          cout<<"\t\t\t";
          displayQueue(que);
          cout<<"\t\t\t";
          cout<<action;
          cout<<"\t\t\t";
          displayStack(out);
          cout<<endl;
}

void input()
{
   string str;
   int n;
```

```
   cout<<"*************** Enter Number of Variables
****************************"<<endl;
   cin>>n;
   cout<<"*************** Enter Productions
************************************"<<endl;
   for(int i=0;i<n;i++)
   {
      cin>>str;
      v.push_back(str);
   }
   cout<<"Enter a input string"<<endl;
   cin>>instr;
}

int findindicesrow(string d)
{
   int c=0;
   for(int i=0;i<=terminal.size();i++)
   {
      if(tab[0][i] == d)
      {
         c = i;
      }
   }
   return c;
}

int findindicescolumn(string d)
{
   int c=0;
   for(int i=0;i<=terminal.size();i++)
   {
      if(tab[i][0] == d)
      {
         c = i;
      }
   }
   return c;
}
```

```
void PostToIn()
{
    string temp;
    while(!out.empty())
    {
        temp = out.top();
        postfix.insert(0,temp);
        out.pop();
    }
    char ch;
    int temp1;
    int first;
    int second;
    int result;
    string op1;
    string op2;
    for(int i=0;i<postfix.length();i++)
    {
        ch = postfix[i];
        if(Isoperand(ch))
        {
            temp = ch;
            stringstream convertch(temp);
            convertch>>temp1;
            poststk.push(temp);
            resultstk.push(temp1);
        }
        else if(Isoperator(ch))
        {
            op1 = poststk.top();
            first = resultstk.top();
            poststk.pop();
            resultstk.pop();
            op2 = poststk.top();
            second = resultstk.top();
            poststk.pop();
            resultstk.pop();
            infix = op2 + ch + op1;
```

```
            poststk.push(infix);
            switch(operation)
            {
            case 1:
                resultstk.push(second + first);
                cout<<"\t\t\t\t\t\t\t\t\t"<<infix<<","<<second << "+" <<first
<<"---->"<<resultstk.top()<<endl;
                break;
            case 2:
                resultstk.push(second - first);
                cout<<"\t\t\t\t\t\t\t\t\t"<<infix<<","<<second << "-" <<first
<<"---->"<<resultstk.top()<<endl;
                break;
            case 3:
                resultstk.push(second * first);
                cout<<"\t\t\t\t\t\t\t\t\t"<<infix<<","<<second << "*" <<first
<<"---->"<<resultstk.top()<<endl;
                break;
            case 4:
                resultstk.push(second / first);
                cout<<"\t\t\t\t\t\t\t\t\t"<<infix<<","<<second << "/" <<first
<<"---->"<<resultstk.top()<<endl;
                break;
            }
        }
    }
    for(int i=0;i<infix.length();i++)
    {
        out.push(infix[i]);
    }
    display("None");
    cout<<"\t\t\t\t\t\t\t\t\t"<<resultstk.top()<<" --Answer"<<endl;
}

void traverse()
{
    string tempqtop;
    string tempstop;
    int row;
```

```
    int column;
    for(int i=0;i< 2* instr.length();i++)
    {
        qtop = que.front();
        if(qtop == '1' || qtop == '2' || qtop == '3' || qtop == '4')
        {
            qtop = 'i';
        }
        tempqtop = qtop;
        stop = stk.top();
        if(stop == '1' || stop == '2' || stop == '3' || stop == '4')
        {
            stop = 'i';
        }
        tempstop = stop;
        column = findindicesrow(tempqtop);
        row = findindicescolumn(tempstop);
        if(tab[row][column] == "<")
        {
            shift();
        }
        else if(tab[row][column] == ">")
        {
            reduce();
        }
        else{
            cout<<"************** ERROR *************"<<endl;
        }
    }
}

void intialize()
{
    cout<<"******************* BUP *******************"<<endl;
    input();
    int i=0;
    while(instr[i]!='\0')
        {
            que.push(instr[i]);
```

```
        i++;
    }
    que.push('$');
    stk.push('$');
}

int main()
{
   while(ch=='Y' || ch=='y')
   {
   intialize();
   for(int i=0;i<v.size();i++)
   {
      vector<string> temp = divideProduction(v[i]);
      for(int j=0 ;j<temp.size();j++) {
      rightp.push_back(temp[j]) ;
      leftp.push_back(v[i][0]) ;
      }
      prod[v[i][0]] = temp ;
   }
   for(int i = v.size() - 1 ; i >= 0 ; i --) {
         if(first.find(v[i][0]) == first.end())
   {
                  firstFinder(v[i][0]) ;
   }
   }
   cout<<"Firsts Are :- "<<endl;

   for(map<char , vector<char> > :: iterator it = first.begin() ; it != first.end() ; it ++)
   {
      vector<char> temp = first[it -> first] ;
      cout << "First(" << it -> first << ") :-" ;
      for(int i = 0 ; i<temp.size() ; i++) {
                  cout << temp[i] ;
                  if(i != temp . size() - 1){
                           cout << " , " ;
                  }
         }
         cout <<endl;
```

```
}
for(int i = v.size() - 1 ; i >= 0 ; i --) {
        if(last.find(v[i][0]) == last.end())
                lastFinder(v[i][0]) ;
}
cout<<"Lasts Are :- "<<endl;

for(map<char , vector<char> > :: iterator it = last.begin() ; it != last.end() ; it ++)
{
    vector<char> temp = last[it -> first] ;
    cout << "Last(" << it -> first << ") :-" ;
    for(int i = 0 ; i<temp.size() ; i++) {
                cout << temp[i] ;
                if(i != temp . size() - 1){
                        cout << " , " ;
                }
        }
        cout <<endl;
}

for(set<char> :: iterator it = ter . begin() ; it != ter . end() ; it ++)
{
    terminal.push_back(*it) ;
}
terminal.push_back('$');
cout<<"Terminal are  -->  ";
for(int i=0;i<terminal.size();i++)
{
    cout<<terminal[i]<<",";
}
cout<<endl;
table();
cout<<"********* Traversing is ... ***********"<<endl;
cout<<"Stack"<<"\t\t\t"<<"Queue"<<"\t\t\t"<<"Action\t\t\t"<<"Output"<<endl<<endl;
display("None");
traverse();
PostToIn();
cout<<"Wnat to parse more strings, please enter Y/y"<<endl;
cin>>ch;
```

```
    if(ch=='y' || ch=='Y')
    {
    system("cls");
    allclear();
    }
    else{
        exit(0);
    }
    }
return 0;
}
```

Output :-

```
"F:\Semester 6\compiler design\Lab\Compiler_Design\CD_Program_9.exe"
******************** BUP ********************
**************** Enter Number of Variables ********************************
3
**************** Enter Productions ****************************************
E-E+T|T
T-T*F|F
F-(E)|i
Enter a input string
1+3*4+2
Firsts Are :-
First(E) :-( , * , + , i
First(F) :-( , i
First(T) :-( , * , i
Lasts Are :-
Last(E) :-) , * , + , i
Last(F) :-) , i
Last(T) :-) , * , i
Terminal are  -->  (,),*,+,i,$,
*******Parse table is ********************


#       (       )       *       +       i       $
(       <       =       <       <       <       -
)       -       >       >       >       -       >
*       <       >       >       >       <       >
+       <       >       <       >       <       >
i       -       >       >       >       -       >
$       <       -       <       <       <       -
```

```
********* Traversing is ... ***********
Stack                   Queue                   Action                  Output

$                       1+3*4+2$                        None
$1                      +3*4+2$                 shift
$                       +3*4+2$                 Reduce                  1
$+                      3*4+2$                  shift                   1
$+3                     *4+2$                   shift                   1
$+                      *4+2$                   Reduce                  13
$+*                     4+2$                    shift                   13
$+*4                    +2$                     shift                   13
$+*                     +2$                     Reduce                  134
$+                      +2$                     Reduce                  134*
$                       +2$                     Reduce                  134*+
$+                      2$                      shift                   134*+
$+2                     $                       shift                   134*+
$+                      $                       Reduce                  134*+2
$                       $                       Reduce                  134*+2+
                                                                        3*4,3*4---->12
                                                                        1+3*4,1+12---->13
                                                                        1+3*4+2,13+2---->15
$                       $                       None                    1+3*4+2
                                                                        15 --Answer
Wnat to parse more strings, please enter Y/y
```