Program No. 10

Aim :- Implement Static LR Parser.

Source Code :-

```
/*
  Lab Name - Compiler Design
  Objective - LR Parser
  Name - Anup Agrawal
  Roll No. - UE143014
  Date - 26/04/2017
*/
#include<bits/stdc++.h>
using namespace std;

vector<string> v;
stack<char> stk;
queue<char> que;
char stop;
char qtop;
string instr;
string tab[20][20];
vector<char> leftp;
vector<string> rightp;

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 divide()
{
   string temp = "";
   for(int i=0;i<v.size();i++)
   {
      temp = v[i];
      temp = temp.substr(2,temp.length() - 2);
      rightp.push_back(temp) ;
      leftp.push_back(v[i][0]) ;
      temp ="";
   }
}

void table()
{
      string tab1[10][8] = {
       {"#" , "(" , ")" , "x" , "," , "$" , "S" , "L"} ,
                  {"0" , "s2" , "_" , "s1" , "_" , "_" , "3" , "_"} ,
                  {"1" ,"r2" , "r2" , "r2" , "r2" , "r2" , "_" , "_"} ,
                  {"2" ,"s2" , "_" , "s1" , "_" , "_" , "6" , "4"} ,
                  {"3" ,"_" , "_" , "_" , "_" , "a" , "_" , "_"} ,
                  {"4" ,"_" , "s5" , "_" , "s7" , "_" , "_" , "_"} ,
                  {"5" ,"r1" , "r1" , "r1" , "r1" , "r1" , "_" , "_"} ,
                  {"6" ,"r3" , "r3" , "r3" , "r3" , "r3" , "_" , "_"} ,
                  {"7" ,"s2" , "_" , "s1" , "_" , "_" , "8" , "_"} ,
                  {"8" ,"r4" , "r4" , "r4" , "r4" , "r4" , "_" , "_"}
         } ;

         for(int i=0;i<10;i++)
   {
      for(int j=0;j<9;j++)
      {
         tab[i][j] = tab1[i][j];
      }
   }
```

```
   cout<<endl;
   cout<<"********* Parse Table **********"<<endl;
         for(int i=0;i<10;i++)
   {
      for(int j=0;j<8;j++)
      {
         cout<<tab[i][j]<<"\t";
      }
      cout<<endl;
   }

}

int findindicesrow(string d)
{
   int c=0;
   for(int i=0;i<=10;i++)
   {
      if(tab[0][i] == d)
      {
         c = i;
      }
   }
   return c;
}

int findindicescolumn(string d)
{
   int c=0;
   for(int i=0;i<=10;i++)
   {
      if(tab[i][0] == d)
      {
         c = i;
      }
   }
   return c;
}
```

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

void shift(string str)
{
   if(que.front() != '$')
   {
      display("Shift",str);
      qtop = que.front();
      stk.push(qtop);
      stk.push(str[1]);
      que.pop();
   }
}

void reduce(string str)
{
   string temp = "";
   string prod;
   string stat;
   string numstate;
   char pushprod;
   char number = str[1];
   int num;
   int r;
   int c;
   int cntpop;
   int cnt = 1;
   string pp ="";
   pp = pp + number;
```

```
    stringstream convertch(pp);
    convertch>>num;
    pp = "";
    string temp1 = ", Prod --> ";
    temp = str +temp1 + v[num];
    display("Reduce",temp);
    prod = rightp[num];
    cntpop = 2 * prod.length();
    for(int i=0;i<cntpop;i++)
    {
       stk.pop();
    }
    string ss = "";
    pushprod = leftp[num];
    ss = ss + pushprod;
    numstate = stk.top();
    r = findindicescolumn(numstate);
    stk.push(pushprod);
    c = findindicesrow(ss);
    ss = "";
    stat = tab[r][c];
    stk.push(stat[0]);
}

void traverse()
{
    cout<<endl<<endl;
    cout<<"********** Parsing Steps ***********"<<endl<<endl;
    cout<<"Stack\t\t\t   Queue\t\t\t       Action\t\t\t       TableEntry"<<endl;
    string tempstop;
    string tempqtop;
    int row =0;
    int column =0;
    while(tab[row][column] != "a")
    {
       stop = stk.top();
       qtop = que.front();
       tempqtop = qtop;
       tempstop = stop;
```

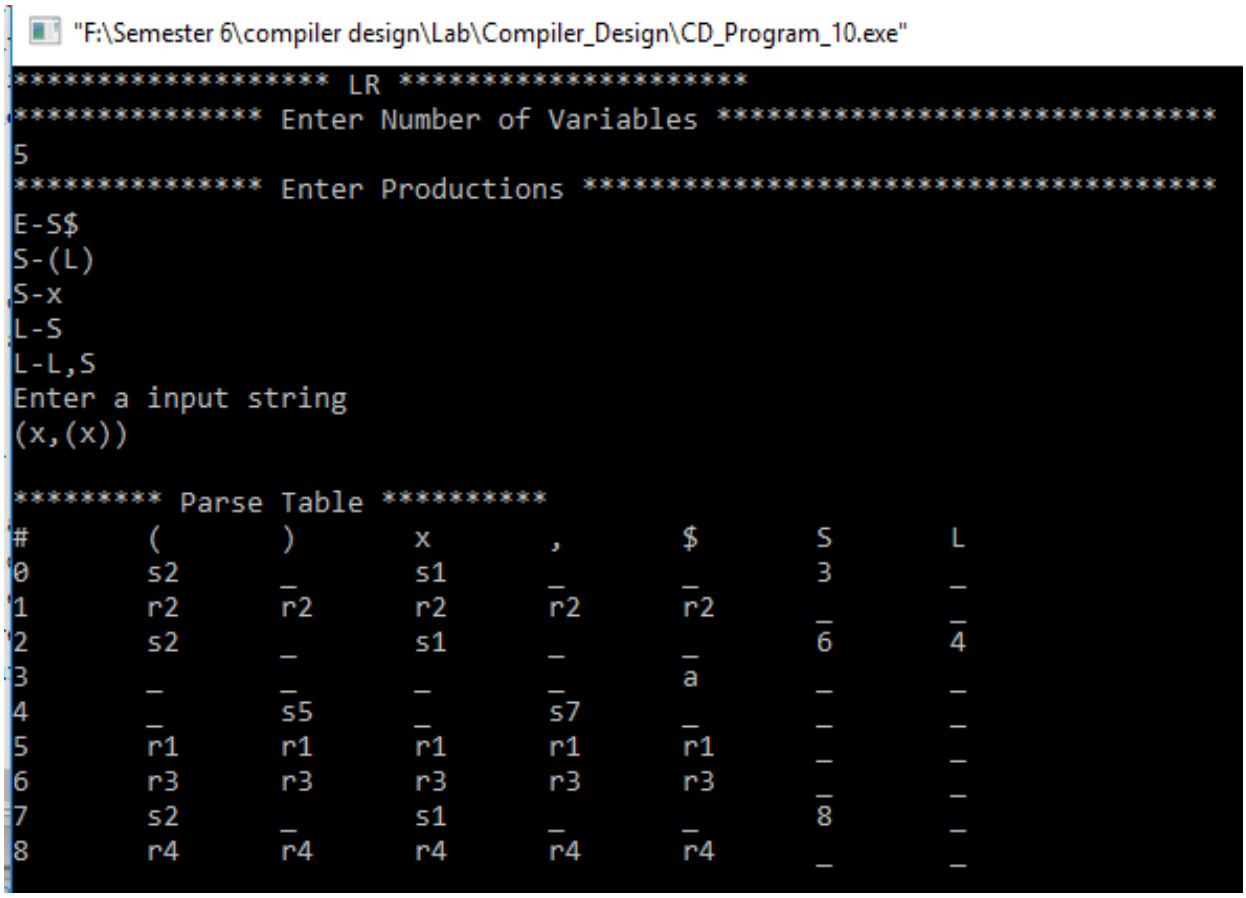
```
        column = findindicesrow(tempqtop);
        row = findindicescolumn(tempstop);
        if(tab[row][column][0] == 's')
        {
           shift(tab[row][column]);
        }
        else if(tab[row][column][0] == 'r')
        {
           reduce(tab[row][column]);
        }
        else if(tab[row][column] == "a")
        {
           display("None","None");
           cout<<"Successfully Parsed ..."<<endl;
        }
        else {
           cout<<"Error"<<endl;
           break;
        }
    }
}

void intialize()
{
   cout<<"******************** LR **********************"<<endl;
   input();
   int i=0;
   while(instr[i]!='\0')
        {
           que.push(instr[i]);
           i++;
        }
        que.push('$');
        stk.push('0');
}

int main()
{
   intialize();
```

```
    divide();
    table();
    traverse();
    return 0;
}
```

Output :-



```
********** Parsing Steps ************

Stack                   Queue                           Action                  TableEntry
0                       (x,(x))$                        Shift                   s2
0(2                     x,(x))$                         Shift                   s1
0(2x1                   ,(x))$                          Reduce                  r2, Prod --> S-x
0(2S6                   ,(x))$                          Reduce                  r3, Prod --> L-S
0(2L4                   ,(x))$                          Shift                   s7
0(2L4,7                 (x))$                           Shift                   s2
0(2L4,7(2                       x))$                    Shift                   s1
0(2L4,7(2x1                     ))$                     Reduce                  r2, Prod --> S-x
0(2L4,7(2S6                     ))$                     Reduce                  r3, Prod --> L-S
0(2L4,7(2L4                     ))$                     Shift                   s5
0(2L4,7(2L4)5                   )$                      Reduce                  r1, Prod --> S-(L)
0(2L4,7S8                       )$                      Reduce                  r4, Prod --> L-L,S
0(2L4                   )$                      Shift                   s5
0(2L4)5                 $                       Reduce                  r1, Prod --> S-(L)
0S3                     $                       None                    None
Successfully Parsed ...

Process returned 0 (0x0)   execution time : 16.631 s
Press any key to continue.
```