```
#include<iostream>
#include<stdlib.h>
#include<fstream>
#include<vector>
#include<string>
#include<stack>
using namespace std;

//---------------------------------------
// CS421 File ll1.cpp for HW3A LL1 Table-Driven Parser
// Your name: Lewis Shine
//---------------------------------------

const int ROW = 3;
const int COL = 2;
vector<char> M[ROW][COL];  // the table of rules :: 3 rows for S, A, B :: 2 rows
for 0, 1 :: Each slot contains a rule's right side :: which is a vector of
characters
stack<char> stacc;

//  ------- conversion functions -----------------------

// to convert non-terms S, A, B to table rows 0, 1, 2
int toRow(char C)
{
  if(C == 'S')
    return 0;
  else if(C == 'A')
    return 1;
  else if(C == 'B')
    return 2;
  else
    {
      cout << "Error: Character (" << C << ") not accepted by function \'toRow'"
<< endl;
      exit(1);
    }
}

// to convert '0' and '1' to table columns 0 and 1
int toCol(char c)
{
  if(c == '0')
    return 0;
  else if(c == '1')
    return 1;
  else
    {
      cout << "Error: Character (" << c << ") not accepted by function \'toCol'"
<< endl;
      exit(1);
    }
}

// to convert row 0, 1, 2 to non-terms S, A and B
char toNonterm(int r)
{
  if(r == 0)
    return 'S';
  else if(r == 1)
    return 'A';
  else if(r == 2)
    return 'B';
  else
```

```
    {
      cout << "Error: Integer (" << r << ") not accepted by
function \'toNonterm'" << endl;
      exit(1);
    }
}

// to display a rule's rhs which is in vector V
void displayVector(vector<char> V)
{
  for(int i=0;i<V.size();i++) //loop to display the contents of the vector
passed
    cout << V[i] << ' ';
  cout << '\t';
}

// to read in the rules into M, make sure ; is not stored
void readrules()
{
  char row,col,tmp; //Variables holding the character values for the rows and
columns
  int iCol,iRow; //Variables holding the integer conversions of the character
variables above ^
  ifstream fin ("rules", ios::in);
  fin >> row >> col; //takes in row (char) and the column (char) to set within
the table
  while(fin)
    {
      iRow = toRow(row);//Calls function to convert row (char) to iRow (int)
      iCol = toCol(col);//Calls function to convert col (char) to iCol (int)
      (M[iRow][iCol]).push_back(col);//adds the first character of the bnf rule
to the vector
      fin >> tmp;//reading bnf rhs rules
      while(tmp != ';')//while loop to get the bnf rule up to the ';'
      {
        (M[iRow][iCol]).push_back(tmp);//adds the next character to the bnf
grammar
        fin >> tmp;//reading bnf rhs rules
      }
      fin >> row >> col; //takes in row (char) and the column (char) to set
within the table
    }
  for(int r=0;r<ROW;r++) //Creates and displays the table
    {
      cout << toNonterm(r) << ":\t";
      for(int c=0;c<COL;c++)
      displayVector(M[r][c]);
      cout << endl;
    }//End of the table
}

//  pushes V contents to the stack
void addtostack(vector<char> V)
{
  cout << "Pushing rhs of a rule to the stack." << endl;
  for(int i=V.size()-1;i>=0;i--)
    stacc.push(V[i]);
}

int main()
{
  readrules();//M is filled and displayed
  string ss;//String for user input
  cout << "Enter a string made of 0's and/or 1's: ";
  cin >> ss;
  if(ss.length() > 4 or ss.length() < 4)
    {
      cout << "Error: String of unacceptable length" << endl;
      exit(1);
    }
  stacc.push('S');//Puts the starting character for the stack to build off of
  int i = 0;//index for ss
  if(stacc.empty() or ss[i] == '\0')
    {
      cout << "Error: immediate failure due to lack of stack contents..." <<
endl;
      exit(1);
    }
  while (ss[i] != '\0')//for each char of ss
    {
      cout << "Stack:" << endl;//Beginning of the display for the Stack with the
starting character
      char cc = stacc.top();//Current Character (cc) gets the top element in the
stack
      stack<char> tmp;
      tmp = stacc;
      stacc.pop();
      for(int s=0;s<stacc.size();s++) //loop to display the contents of the
vector passed
      {
        cout << tmp.top() << endl;
        tmp.pop();
      }
      cout << "--------------------------------\nCurrent Character is: " <<
ss[i] << endl;
      if(cc == 'S' or cc == 'A')
      addtostack(M[toRow(cc)][toCol(ss[i])]);
      else if(ss[i] == cc)
      {
        cout << "Match!" << endl;
        i++;
      }
      else
      {
        cout << "Error: immediate failure due to mismatch between stack
character (" << cc << ") and user input character (" << ss[i] << endl;
        exit(1);
      }
      cout << endl;
    }
  cout << "This string has been accepted!" << endl;
  return 0;
}// end of main
```