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

void printTable(map<char, map<char, string>> &table, vector<char> &terminals, vector<char>
&nonTerminals)
{
    cout << "\nLL(1) Parsing Table:\n";
    cout << setw(6) << " ";

    for (auto &t : terminals)
        cout << setw(6) << t;
    cout << endl;

    for (auto &nt : nonTerminals)
    {
        cout << setw(6) << nt;
        for (auto &t : terminals)
        {
            if (table[nt].find(t) != table[nt].end())
                cout << setw(6) << table[nt][t];
            else
                cout << setw(6) << " ";
        }
        cout << endl;
    }
}

int main()
{
    map<char, vector<string>> grammar;
    int ruleCount;
    char startSymbol, epsilon;

    cout << "Enter the epsilon character (e.g. '#' or 'e' for empty productions): ";
    cin >> epsilon;

    cout << "How many grammar rules: ";
    cin >> ruleCount;

    cout << "Enter the start symbol: ";
    cin >> startSymbol;

    cout << "Enter the grammar (e.g. E→E+T|T):\n";
    for (int i = 0; i < ruleCount; i++)
    {
        char nonTerminal;
        string production;

        cout << "Enter non-terminal: ";
        cin >> nonTerminal;
        cout << nonTerminal << "→";
        cin >> production;

        stringstream ss(production);
        string prod;
        while (getline(ss, prod, '|'))
            grammar[nonTerminal].push_back(prod);
    }

    set<char> terminalsSet, nonTerminalsSet;

    for (auto &rule : grammar)
    {
        nonTerminalsSet.insert(rule.first);
        for (const auto &prod : rule.second)
        {
            for (char ch : prod)
            {
                if (!isupper(ch) && ch != epsilon)
                    terminalsSet.insert(ch);
```

```
            }
        }
    }
    terminalsSet.insert('$');

    vector<char> terminals(terminalsSet.begin(), terminalsSet.end());
    vector<char> nonTerminals(nonTerminalsSet.begin(), nonTerminalsSet.end());

    map<char, map<char, string>> table;

    for (auto &rule : grammar)
    {
        char nonTerminal = rule.first;
        for (const auto &prod : rule.second)
        {
            set<char> firstSet;
            if (isupper(prod[0]))
                first(grammar, prod[0], firstSet, epsilon);
            else
                firstSet.insert(prod[0]);

            for (char terminal : firstSet)
            {
                if (terminal != epsilon)
                    table[nonTerminal][terminal] = prod;
                else
                {
                    set<char> followSet;
                    set<char> visited;
                    follow(grammar, nonTerminal, followSet, startSymbol, visited, epsilon);
                    for (char followChar : followSet)
                        table[nonTerminal][followChar] = prod;
                    if (followSet.find('$') != followSet.end())
                        table[nonTerminal]['$'] = prod;
                }
            }
        }
    }

    printTable(table, terminals, nonTerminals);

    return 0;
}

OUTPUT:

Enter the epsilon character (e.g. '#' or 'e' for empty productions): ε
How many grammar rules: 3
Enter the start symbol: S
Enter the grammar (e.g. E→E+T|T):
Enter non-terminal: S
S→AB
Enter non-terminal: A
A→a|ε
Enter non-terminal: B
B→b|ε

LL(1) Parsing Table:
           $     a     b
     A     ε     a     ε
     B     ε           b
     S    AB    AB
```