```cpp
#include <iostream>
#include <string>
#include <sstream>
#include <vector>
#include <map>

using namespace std;

int main()
{
	map<int, string> grammar_productions;

	string all_term = "";

	int total_productions=0;
	cout << "Enter the total number of productions : ";
	cin >> total_productions;

	cout << endl;

	for (int i = 0; i < total_productions; ++i)
	{
		cout << "Enter the production numbered [" << i+1 << "] : ";
		cin >> grammar_productions[i+1];
		for(int j = 2; j < grammar_productions[i+1].size(); j++)
		{
			if(!((int)grammar_productions[i+1][j] >= 65 && (int)grammar_productions[i+1]
[j] <= 90))
			{
				stringstream ss;
				ss << grammar_productions[i+1][j];
				all_term.append(ss.str());
			}
		}
	}
	all_term.append("$");


	string all_n_term = "";

	map<int, string>::iterator it;
	cout << "\nAll production of the grammar are : \n";
	for(it = grammar_productions.begin(); it != grammar_productions.end(); it++)
	{
		cout << "(" << it->first << ") " << it->second << endl;
		string temp = it->second;

		for(int j = 0; j < temp.size(); j++)
		{
			unsigned found = all_n_term.find(temp[j]);
			if(found == string::npos && (((int)temp[j] >= 65 && (int)temp[j] <= 90)))
			{
				stringstream ss;
				string temp_n_term;
				ss << temp[j];
				ss >> temp_n_term;
				all_n_term.append(temp_n_term);
			}
		}
	}
	cout << endl;

	cout << "All Terminals : ";
	for (int i = 0; i < all_term.size(); ++i)
	{
		cout << all_term[i];
	}
	cout << endl;

	cout << "All Non Terminals : ";
	for (int i = 0; i < all_n_term.size(); ++i)
	{
```

```
        cout << all_n_term[i];
}
cout << endl;

int total_states;
cout << "Enter the total number of states : ";
cin >> total_states;
cout << endl;

//ACTION table

map<string, string> action_entry;
for(int i = 0; i < total_states; i++)
{
        for(int j = 0; j < all_term.size(); j++)
        {
                stringstream ss;
                ss << i << all_term[j];
                string mtrx_index;
                mtrx_index = ss.str();
                cout << "[" << i << "][" << all_term[j] << "] (# for error entry) :";
                cin >> action_entry[mtrx_index];
        }
}
cout << "\nACTION\n";
map<string, string>::iterator it_2;
for(it_2 = action_entry.begin(); it_2 != action_entry.end(); it_2++)
        cout << it_2->first << "        " << it_2->second << endl;


//GOTO table

map<string, string> goto_entry;
for(int i = 0; i < total_states; i++)
{
        for(int j = 0; j < all_n_term.size(); j++)
        {
                stringstream ss;
                ss << i << all_n_term[j];
                string mtrx_index;
                mtrx_index = ss.str();
                cout << "[" << i << "][" << all_n_term[j] << "] (# for error entry) : ";
                cin >> goto_entry[mtrx_index];
        }
}
cout << "\nGOTO\n";
map<string, string>::iterator it_3;
for(it_3 = goto_entry.begin(); it_3 != goto_entry.end(); it_3++)
        cout << it_3->first << "        " << it_3->second << endl;


//parsing
string input_string;
cout << "Enter the string to be parsed : ";
cin >> input_string;
input_string.append("$");
int ip=0;

vector<string> stack;
stack.push_back("0");
string stack_back = stack.back();

while(true)
{
        stringstream ss;
        ss << stack_back << input_string[ip];

        string action_table_index = ss.str();
        string action_table_entry = action_entry[action_table_index];

        if(action_table_entry == "acc")
        {
```

```
                        cout << "ACCEPT" << endl;
                        break;
                }
                else if(action_table_entry[0] == 's')   //SHIFT
                {
                        cout << "SHIFT" << endl;
                        ss.str("");
                        ss.clear();
                        ss << input_string[ip];
                        stack.push_back(ss.str());
                        stack.push_back(action_table_entry.substr(1,(action_table_entry.size()-1)));
                        stack_back = stack.back();
                        ip++;
                }
                else if(action_table_entry[0] == 'r')   //REDUCE
                {
                        int prod_number;

                        ss.str("");
                        ss.clear();
                        ss << action_table_entry.substr(1,(action_table_entry.size()-1));
                        ss >> prod_number;

                        string temp_prod = grammar_productions[prod_number];

                        for (int i = 0; i < (temp_prod.size()-2)*2; ++i)
                        {
                                stack.pop_back();
                        }
                        stack_back = stack.back();

                        ss.str("");
                        ss.clear();

                        ss << stack_back << temp_prod[0];
                        string goto_table_entry = goto_entry[ss.str()];

                        ss.str("");
                        ss.clear();
                        ss << temp_prod[0];
                        stack.push_back(ss.str());
                        stack.push_back(goto_table_entry);
                        stack_back = stack.back();

                        cout << "REDUCE by " << temp_prod << endl;
                }
                else
                {
                        cout << "ERROR!!!\nCan't parse the given string.\n" << endl;
                        break;
                }
                stack_back = stack.back();
        }



        return 0;
}
```