Aim: - Write a c program to implement simple lr parsing algorithm.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX 100 // Maximum number of productions and input length

typedef struct {

    char lhs;            // Left-hand side of the production

    char rhs[MAX];      // Right-hand side of the production

} Production;

typedef struct {

    char action[MAX][MAX]; // Action table

    int gotoTable[MAX][MAX]; // Goto table

} ParsingTable;

// Grammar definitions

// Production productions[MAX];

// int productionCount = 0;

// Function to initialize parsing table

void initializeParsingTable(ParsingTable \*table) {

    for (int i = 0; i < MAX; i++) {

        for (int j = 0; j < MAX; j++) {

            table->action[i][j] = ' '; // Initialize with spaces

            table->gotoTable[i][j] = -1; // Initialize with -1

        }

    }

}

// Function to define the LR parsing table (hardcoded for this example)

void defineParsingTable(ParsingTable \*table) {

    // Action table for grammar S -> aS | b

    table->action[0][0] = 's'; // state 0, input 'a' -> shift to state 2

    table->action[0][1] = 's'; // state 0, input 'b' -> shift to state 1

    table->action[1][0] = 'r'; // state 1, input 'a' -> reject

    table->action[1][1] = 'e'; // state 1, input 'b' -> error/reject

    table->action[1][2] = 'a'; // state 1, end of input -> accept

    table->action[2][0] = 's'; // state 2, input 'a' -> shift

    table->action[2][1] = 's'; // state 2, input 'b' -> shift

    // Goto table with complete transitions

    // Define goto transitions

    table->gotoTable[0][0] = 2; // goto state 2 on 'a'

    table->gotoTable[0][1] = 1; // goto state 1 on 'b'

    table->gotoTable[2][1] = 1; // on 'b' goto state 1

}

// Function to perform LR parsing

void parseInput(const char \*input, ParsingTable \*table) {

    int stateStack[MAX];

    char symbolStack[MAX];

    int top = 0;

    stateStack[0] = 0;

    int i = 0;

    printf("\nParsing Steps:\n");

    printf("Stack\t\tInput\t\tAction\n");

    while (1) {

        printf("State: %d\t\t%s\t\t", stateStack[top], &input[i]);

        char currentSymbol;

        if (input[i] != '\0') {

            currentSymbol = input[i];

        } else {

            currentSymbol = '$';

        }

        int currentState = stateStack[top];

        int symbolIndex = (currentSymbol == '$') ? 2 : (currentSymbol - 'a');

        char action = table->action[currentState][symbolIndex];

        if (action == 's') {

            if (i >= strlen(input)) {

                printf("Error\n\nInput string rejected!\n");

                return;

            }

            stateStack[++top] = table->gotoTable[currentState][currentSymbol - 'a'];

            symbolStack[top] = currentSymbol;

            i++;

            printf("Shift\n");

        }

        else if (action == 'r') {

            if (top <= 0) {

                printf("Error\n\nInput string rejected!\n");

                return;

            }

            top--;

            printf("Reduce\n");

        }

        else if (action == 'a') {

            printf("Accept\n\nInput string accepted!\n");

            return;

        }

        else {

            printf("Error\n\nInput string rejected!\n");

            return;

        }

    }

}

// Function to add a production to the grammar

int main() {

    ParsingTable table;

    initializeParsingTable(&table);

    defineParsingTable(&table);

    char input[MAX];

    printf("Enter the input string (e.g., ab): ");

    scanf("%s", input);

    parseInput(input, &table);

    return 0;

}

Output: -













