8.Write a C program to find FOLLOW( ) - predictive parser for the given grammar  
S → AaAb / BbBa  
A → ∈  
B → ∈

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define MAX\_NON\_TERMINALS 10

#define MAX\_PRODUCTIONS 10

#define MAX\_RULE\_LENGTH 10

// Struct to store productions

typedef struct {

char nonTerminal;

char production[MAX\_RULE\_LENGTH][MAX\_RULE\_LENGTH];

int numProds;

} Grammar;

// Function to find FOLLOW sets for a given grammar

void findFollow(Grammar \*grammar, char nonTerminals[], int numNonTerminals, char startSymbol) {

// Initialize FOLLOW sets

char follow[MAX\_NON\_TERMINALS][MAX\_RULE\_LENGTH];

for (int i = 0; i < numNonTerminals; i++) {

follow[i][0] = '\0'; // Empty FOLLOW set

}

// Add $ (end of input) to the FOLLOW set of the start symbol

follow[0][0] = '$'; // Assuming start symbol is first non-terminal (S)

int changes;

do {

changes = 0;

// Process each production

for (int i = 0; i < numNonTerminals; i++) {

for (int j = 0; j < grammar[i].numProds; j++) {

char\* prod = grammar[i].production[j];

// Traverse production from left to right

for (int k = 0; k < strlen(prod); k++) {

// If the symbol is a non-terminal, check what follows it

if (prod[k] >= 'A' && prod[k] <= 'Z') {

int index = prod[k] - 'A';

// Check if epsilon (empty production) is there

if (prod[k + 1] == '\0') {

// Add FOLLOW of the current production to the FOLLOW set

int len = strlen(follow[index]);

for (int l = 0; l < len; l++) {

if (strchr(follow[i], follow[index][l]) == NULL) {

follow[i][strlen(follow[i])] = follow[index][l];

changes = 1;

}

}

}

}

}

}

}

} while (changes);

// Print the FOLLOW sets

for (int i = 0; i < numNonTerminals; i++) {

printf("FOLLOW(%c) = { ", nonTerminals[i]);

for (int j = 0; follow[i][j] != '\0'; j++) {

printf("%c ", follow[i][j]);

}

printf("}\n");

}

}

int main() {

Grammar grammar[MAX\_NON\_TERMINALS];

char nonTerminals[MAX\_NON\_TERMINALS] = {'S', 'A', 'B'};

int numNonTerminals = 3;

// Defining the grammar based on the user's input

grammar[0].nonTerminal = 'S';

grammar[0].numProds = 2;

strcpy(grammar[0].production[0], "AaAb");

strcpy(grammar[0].production[1], "BbBa");

grammar[1].nonTerminal = 'A';

grammar[1].numProds = 1;

strcpy(grammar[1].production[0], "?");

grammar[2].nonTerminal = 'B';

grammar[2].numProds = 1;

strcpy(grammar[2].production[0], "?");

// Find and display the FOLLOW sets for the grammar

findFollow(grammar, nonTerminals, numNonTerminals, 'S');

return 0;

}

