CODE:

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

Code:

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define MAX 10

char productions[MAX][MAX], follow[MAX][MAX];

int prod\_count;

void addToSet(char \*set, char val) {

if (strchr(set, val) == NULL) {

int len = strlen(set);

set[len] = val;

set[len + 1] = '\0';

}

}

void computeFollow(char non\_terminal, char \*res) {

if (productions[0][0] == non\_terminal)

addToSet(res, '$'); // Rule 1: FOLLOW(S) = { $ }

for (int i = 0; i < prod\_count; i++) {

for (int j = 2; j < strlen(productions[i]); j++) {

if (productions[i][j] == non\_terminal) {

if (productions[i][j + 1] != '\0') {

if (!isupper(productions[i][j + 1]))

addToSet(res, productions[i][j + 1]); // Terminal follows directly

else {

if (productions[i][j + 1] == 'A' || productions[i][j + 1] == 'B')

computeFollow(productions[i][0], res);

}

} else {

computeFollow(productions[i][0], res); // Inherit FOLLOW from LHS

}

}

}

}

}

int main() {

printf("Enter number of productions: ");

scanf("%d", &prod\_count);

printf("Enter productions (e.g., S=AaAb | BbBa):\n");

for (int i = 0; i < prod\_count; i++)

scanf("%s", productions[i]);

for (int i = 0; i < prod\_count; i++) {

follow[i][0] = '\0';

computeFollow(productions[i][0], follow[i]);

}

printf("\nFOLLOW sets:\n");

for (int i = 0; i < prod\_count; i++)

printf("FOLLOW(%c) = {%s}\n", productions[i][0], follow[i]);

return 0;

}

OUTPUT:

