// C program to calculate the First and

// Follow sets of a given grammar

#include<stdio.h>

#include<ctype.h>

#include<string.h>

// Functions to calculate Follow

void followfirst(char, int, int);

void follow(char c);

// Function to calculate First

void findfirst(char, int, int);

int count, n = 0;

// Stores the final result

// of the First Sets

char calc\_first[10][100];

// Stores the final result

// of the Follow Sets

char calc\_follow[10][100];

int m = 0;

// Stores the production rules

char production[10][10];

char f[10], first[10];

int k;

char ck;

int e;

void read\_productions()

{

int i;

printf("Enter the number of productions\n");

scanf("%d",&count);

printf("Enter each production in the form X=YZ for X->YZ, and # for epsilon\n");

for(i=0;i<count;i++)

{

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

}

}

int main()

{

int jm = 0;

int km = 0;

int i, choice;

char c, ch;

read\_productions();

int l;

char done\_first[count];

int ptr = -1;

// Initializing the calc\_first array

for(k = 0; k < count; k++) {

for(l = 0; l < 100; l++) {

calc\_first[k][l] = '!';

}

}

int point1 = 0, point2, flag;

for(k = 0; k < count; k++)

{

c = production[k][0];

point2 = 0;

flag = 0;

// Checking if First of c has

// already been calculated

for(l = 0; l <= ptr; l++)

if(c == done\_first[l])

flag = 1;

if (flag == 1)

continue;

// Function call

findfirst(c, 0, 0);

ptr += 1;

// Adding c to the calculated list

done\_first[ptr] = c;

printf("\n First(%c) = { ", c);

calc\_first[point1][point2++] = c;

// Printing the First Sets of the grammar

for(i = 0 + jm; i < n; i++) {

int lark = 0, chk = 0;

for(lark = 0; lark < point2; lark++) {

if (first[i] == calc\_first[point1][lark])

{

chk = 1;

break;

}

}

if(chk == 0)

{

printf("%c, ", first[i]);

calc\_first[point1][point2++] = first[i];

}

}

printf("}\n");

jm = n;

point1++;

}

printf("\n");

printf("-----------------------------------------------\n\n");

char done\_follow[count];

ptr = -1;

// Initializing the calc\_follow array

for(k = 0; k < count; k++) {

for(l = 0; l < 100; l++) {

calc\_follow[k][l] = '!';

}

}

point1 = 0;

int land = 0;

for(e = 0; e < count; e++)

{

ck = production[e][0];

point2 = 0;

flag = 0;

// Checking if Follow of ck

// has already been calculated

for(l = 0; l <= ptr; l++)

if(ck == done\_follow[l])

flag = 1;

if (flag == 1)

continue;

land += 1;

// Function call

follow(ck);

ptr += 1;

// Adding ck to the calculated list

done\_follow[ptr] = ck;

printf(" Follow(%c) = { ", ck);

calc\_follow[point1][point2++] = ck;

// Printing the Follow Sets of the grammar

for(i = 0 + km; i < m; i++) {

int lark = 0, chk = 0;

for(lark = 0; lark < point2; lark++)

{

if (f[i] == calc\_follow[point1][lark])

{

chk = 1;

break;

}

}

if(chk == 0)

{

printf("%c, ", f[i]);

calc\_follow[point1][point2++] = f[i];

}

}

printf(" }\n\n");

km = m;

point1++;

}

}

void follow(char c)

{

int i, j;

// Adding "$" to the follow

// set of the start symbol

if(production[0][0] == c) {

f[m++] = '$';

}

for(i = 0; i < 10; i++)

{

for(j = 2;j < 10; j++)

{

if(production[i][j] == c)

{

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

{

// Calculate the first of the next

// Non-Terminal in the production

followfirst(production[i][j+1], i, (j+2));

}

if(production[i][j+1]=='\0' && c!=production[i][0])

{

// Calculate the follow of the Non-Terminal

// in the L.H.S. of the production

follow(production[i][0]);

}

}

}

}

}

void findfirst(char c, int q1, int q2)

{

int j;

// The case where we

// encounter a Terminal

if(!(isupper(c))) {

first[n++] = c;

}

for(j = 0; j < count; j++)

{

if(production[j][0] == c)

{

if(production[j][2] == '#')

{

if(production[q1][q2] == '\0')

first[n++] = '#';

else if(production[q1][q2] != '\0'

&& (q1 != 0 || q2 != 0))

{

// Recursion to calculate First of New

// Non-Terminal we encounter after epsilon

findfirst(production[q1][q2], q1, (q2+1));

}

else

first[n++] = '#';

}

else if(!isupper(production[j][2]))

{

first[n++] = production[j][2];

}

else

{

// Recursion to calculate First of

// New Non-Terminal we encounter

// at the beginning

findfirst(production[j][2], j, 3);

}

}

}

}

void followfirst(char c, int c1, int c2)

{

int k;

// The case where we encounter

// a Terminal

if(!(isupper(c)))

f[m++] = c;

else

{

int i = 0, j = 1;

for(i = 0; i < count; i++)

{

if(calc\_first[i][0] == c)

break;

}

//Including the First set of the

// Non-Terminal in the Follow of

// the original query

while(calc\_first[i][j] != '!')

{

if(calc\_first[i][j] != '#')

{

f[m++] = calc\_first[i][j];

}

else

{

if(production[c1][c2] == '\0')

{

// Case where we reach the

// end of a production

follow(production[c1][0]);

}

else

{

// Recursion to the next symbol

// in case we encounter a "#"

followfirst(production[c1][c2], c1, c2+1);

}

}

j++;

}

}

}