Name: Atandrit Chatterjee

Reg. No.: RA2011031010042

**Experiment-5** **Computation of FIRST () and FOLLOW ()**

Aim:

To write a program to compute the FIRST() and FOLLOW().

Procedure:

a. For computing the first:

1. If X is a terminal then FIRST(X) = {X}

Example: F -> I | id

We can write it as FIRST(F) -> { ( , id )

2. If X is a non-terminal like E -> T then to get FIRSTI substitute T with other productions

until you get a terminal as the first symbol

3. If X -> ε then add ε to FIRST(X).

b. For computing the follow:

1. Always check the right side of the productions for a non-terminal, whose FOLLOW set is

being found. (never see the left side).

2. (a) If that non-terminal (S,A,B...) is followed by any terminal (a,b...,\*,+,(,)...) , then add that terminal into the FOLLOW set.

(b) If that non-terminal is followed by any other non-terminal then add FIRST of other nonterminal into the FOLLOW set.

Code:

#include <bits/stdc++.h>

int n, m = 0, p, i = 0, j = 0;

char a[10][10], f[10];

void follow(char c);

void first(char c);

int main()

{

system("cls");

int i, z;

char c, ch;

printf("Enter the no of prooductions:\n");

scanf("%d", &n);

printf("Enter the productions:\n");

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

scanf("%s%c", a[i], &ch);

do

{

m = 0;

printf("Enter the elemets whose fisrt & follow is to be found:");

scanf("%c", &c);

first(c);

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

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

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

printf("}\n");

strcpy(f, " ");

m = 0;

follow(c);

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

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

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

printf("}\n");

printf("Continue(0/1)?");

scanf("%d%c", &z, &ch);

} while (z == 1);

system("pause");

return (0);

}

void first(char c)

{

int k;

if (!isupper(c))

f[m++] = c;

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

{

if (a[k][0] == c)

{

if (a[k][2] == '$')

follow(a[k][0]);

else if (islower(a[k][2]))

f[m++] = a[k][2];

else

first(a[k][2]);

}

}

}

void follow(char c)

{

if (a[0][0] == c)

f[m++] = '$';

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

{

for (j = 2; j < strlen(a[i]); j++)

{

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

{

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

first(a[i][j + 1]);

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

follow(a[i][0]);

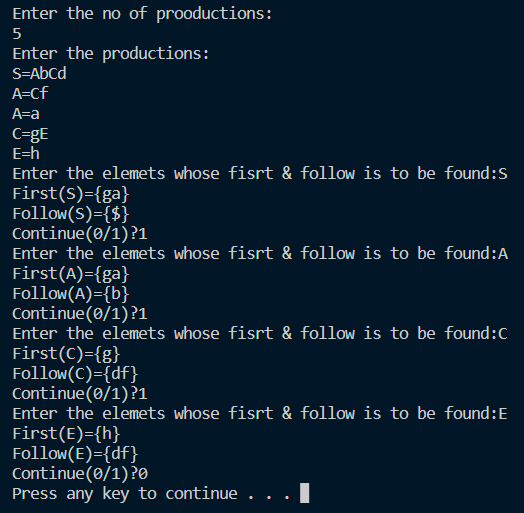
}

}

}

}

Output:



Result:

The FIRST and FOLLOW sets of the non-terminals of a grammar were found  
successfully.