Name: Gokulnath M Prabhu
Class: CS7B
Roll No: 21

Lab Cycle 3 - Experiment 14

Write a program to find First and Follow of any given grammar.

Code:

```c
#include <stdio.h>
#include <string.h>

int n;

char prods[50][50];
char firsts[26][50];
int is_first_done[26];
char follows[26][50];
int is_follow_done[26];

int isTerminal(char c)
{
   if (c < 65 || c > 90)
       return 1;
   return 0;
}
void first(char nonterm)
{
   int index = 0;
   char curr_firsts[50];
   for (int i = 0; i < n; i++)
   {
       if (prods[i][0] == nonterm)
       {
           int curr_prod_index = 2;
           int flag = 0;
           while (prods[i][curr_prod_index] != '\0' && flag == 0)
           {
               flag = 1;
               if (isTerminal(prods[i][curr_prod_index]))
               {
                   curr_firsts[index] = prods[i][2];
                   index++;
                   break;
               }
               if (!is_first_done[prods[i][curr_prod_index] - 65])
```

```
                first(prods[i][curr_prod_index]);
            int in = 0;
            while (firsts[prods[i][curr_prod_index] - 65][in] !=
'\0')
            {
                curr_firsts[index] =
firsts[prods[i][curr_prod_index] - 65][in];
                if (firsts[prods[i][curr_prod_index] - 65][in] ==
'e')
                {
                    curr_prod_index++;
                    flag = 0;
                }
                index++;
                in++;
            }
        }
    }
    curr_firsts[index] = '\0';
    index++;
    strcpy(firsts[nonterm - 65], curr_firsts);
    is_first_done[nonterm - 65] = 1;
}
void follow(char nonterm)
{
    int index = 0;
    char curr_follows[50];
    if (nonterm == prods[0][0])
    {
        curr_follows[index] = '$';
        index++;
    }
    for (int j = 0; j < n; j++)
    {
        int k = 2;
        int include_lhs_flag;
        while (prods[j][k] != '\0')
        {
            include_lhs_flag = 0;
            if (prods[j][k] == nonterm)
            {
                if (prods[j][k + 1] != '\0')
```

```
                {
                    if (isTerminal(prods[j][k + 1]))
                    {
                        curr_follows[index] = prods[j][k + 1];
                        index++;
                        break;
                    }
                    int in = 0;
                    while (firsts[prods[j][k + 1] - 65][in] != '\0')
                    {
                        if (firsts[prods[j][k + 1] - 65][in] == 'e')
                        {
                            include_lhs_flag = 1;
                            in++;
                            continue;
                        }
                        int temp_flag = 0;
                        for (int z = 0; z < index; z++)
                            if (firsts[prods[j][k + 1] - 65][in] ==
curr_follows[z])
                            {
                                temp_flag = 1;
                                in++;
                                break;
                            }
                        if (temp_flag)
                            continue;
                        curr_follows[index] = firsts[prods[j][k + 1] -
65][in];
                        index++;
                        in++;
                    }
                }
                if (prods[j][k + 1] == '\0' || include_lhs_flag == 1)
                {
                    if (prods[j][0] != nonterm)
                    {
                        if (!is_follow_done[prods[j][0] - 65])
                            follow(prods[j][0]);
                        int x = 0;
                        while (follows[prods[j][0] - 65][x] != '\0')
                        {
                            int temp_flag = 0;
```

```
                            for (int z = 0; z < index; z++)
                                if (follows[prods[j][0] - 65][x] ==
curr_follows[z])
                                {
                                    temp_flag = 1;
                                    x++;
                                    break;
                                }
                            if (temp_flag)
                                continue;
                            curr_follows[index] = follows[prods[j][0] -
65][x];
                            index++;
                            x++;
                        }
                    }
                }
            }
            k++;
        }
    }
    curr_follows[index] = '\0';
    index++;
    strcpy(follows[nonterm - 65], curr_follows);
    is_follow_done[nonterm - 65] = 1;
}
int main()
{
    printf("Enter the number of productions: ");
    scanf("%d", &n);
    printf("Enter the productions: \n");
    for (int i = 0; i < n; i++)
        scanf("%s", prods[i]);
    for (int i = 0; i < 26; i++)
        is_first_done[i] = 0;
    for (int i = 0; i < n; i++)
        if (is_first_done[prods[i][0] - 65] == 0)
            first(prods[i][0]);
    for (int i = 0; i < n; i++)
        if (is_follow_done[prods[i][0] - 65] == 0)
            follow(prods[i][0]);
    printf("Firsts:\n");
    for (int i = 0; i < 26; i++)
```

```
        if (is_first_done[i])
            printf("%c : %s\n", i + 65, firsts[i]);
    printf("Follows:\n");
    for (int i = 0; i < 26; i++)
        if (is_follow_done[i])
            printf("%c : %s\n", i + 65, follows[i]);
}
```

Output:

```
→  First_And_Follow git:(master) ✗ ./a.out
Enter the number of productions: 8
Enter the productions:
E=TR
R=+TR
R=e
T=FY
Y=*FY
Y=e
F=(E)
F=i
Firsts:
E : (i
F : (i
R : +e
T : (i
Y : *e
Follows:
E : $)
F : *+$)
R : $)
T : +$)
Y : +$)
→  First_And_Follow git:(master) ✗ _
```