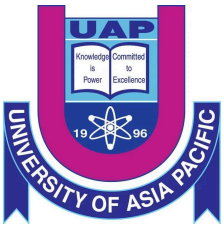


University of Asia Pacific

Department of Computer Science & Engineering

Course code: CSE 430

Course Title: Compiler Design Lab

Submitted To	**Submitted By**
Baivab Das Lecturer, CSE, UAP	Asma Sultana Id: 20101084 Section: B

Lab 4

Problem

FIRST & FOLLOW Function

Description of the problem

The construction of a predictive parser is aided by two functions associated with a grammar G. These functions, FIRST and FOLLOW, allow us to fill in the entries of a predictive parsing table for G, whenever possible. First and Follow sets are needed so that the parser can properly apply the needed production rule at the correct position.

Code

```
#include<bits/stdc++.h>
#include <stdio.h>
#include <string.h>
using namespace std;

void findfirst(char, int, int);
void followfirst(char, int, int);
void follow(char c);


char store_first[10][100];
char store_follow[10][100];
int m = 0, n = 0;
char production[10][10];
char f[10], first[10];
int k,e;
char ck;

int cnt ;

void follow(char c)
{
    int i, j;
```

```
    if (production[0][0] == c)
    {
        f[m++] = '$';       //$ assign
    }
    for (i = 0; i < 10; i++)
    {
        for (j = 2; j < 10; j++)
        {
            if (production[i][j] == c)
            {
                if (production[i][j + 1] != '\0')
                {
                    followfirst(production[i][j + 1], i, j + 2);
                }

                if (production[i][j + 1] == '\0' && c != production[i][0])
                {
                    follow(production[i][0]); //non-terminal production
find: T=
                }
            }
        }
    }
}

void findfirst(char c, int q1, int q2)
{
    int j;
    if (!(isupper(c)))
    {
        first[n++] = c;
    }
    for (j = 0; j < cnt; 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))
                {
                    findfirst(production[q1][q2], q1,(q2 + 1)); //
                }
                else
                    first[n++] = '#';
            }
            else if (!isupper(production[j][2]))
            {
                first[n++] = production[j][2];
            }
            else
            {
                findfirst(production[j][2], j, 3);
            }
        }
    }
}

void followfirst(char c, int c1, int c2)
{
    int k;
    if (!(isupper(c)))
        f[m++] = c;
    else
    {
        int i = 0, j = 1;
        for (i = 0; i < cnt; i++)
        {
            if (store_first[i][0] == c)
                break;
        }
        while (store_first[i][j] != '!') //non-terminal in follow of
original query
        {
            if (store_first[i][j] != '#')
            {
                f[m++] = store_first[i][j];
            }
```

```
            else
            {
                if (production[c1][c2] == '\0')
                {
                    follow(production[c1][0]);  // end of a production
                }
                else
                {
                    followfirst(production[c1][c2], c1, c2 + 1);
                }
            }
            j++;
        }
    }
}



int main()
{
    int jm = 0, km = 0;
    int i, choice;
    char c, ch;

    ifstream inputFile("input.txt");
    if (inputFile.is_open())
    {
        while (inputFile >> production[cnt])
        {
            cnt++;
        }
        inputFile.close();
    }
    else
    {
        cout << "No such file to open..." << endl;
        return 1;
    }

    cout << "Production rules read from textfile: " << endl;
```

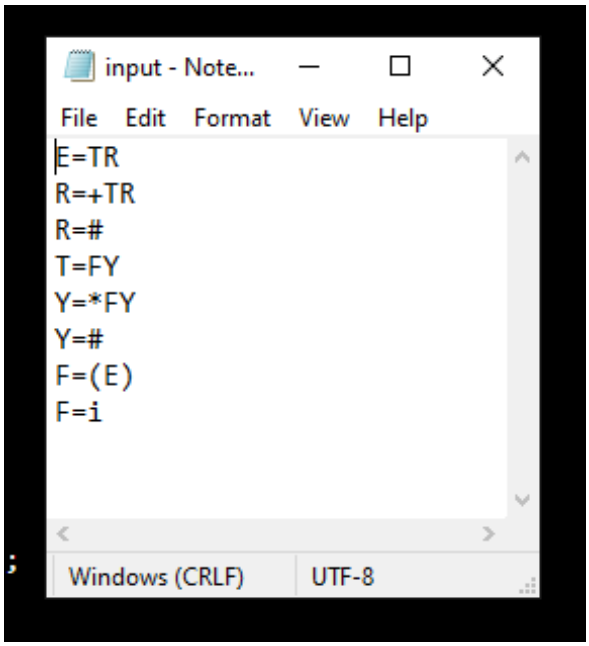
```
    for (int i = 0; i < cnt; i++)
    {
        cout << "Production " << i + 1 << "= " << production[i] << endl;
    }

    int kay;
    char done[cnt];
    int ptr = -1;

    for (k = 0; k < cnt; k++)       //calculate first set
    {
        for (kay = 0; kay < 100; kay++)
        {
            store_first[k][kay] = '!';
        }
    }
    int point1 = 0, point2, x;

    for (k = 0; k < cnt; k++)
    {
        c = production[k][0];   //assigns the first symbol
        point2 = 0;
        x = 0;

        for (kay = 0; kay <= ptr; kay++)    //check if first of c has
already been calculated
            if (c == done[kay])
                x = 1;

        if (x == 1)
            continue;

        //function call
        findfirst(c, 0, 0);
        ptr += 1;

        //adding c to the calculated list
        done[ptr] = c;
        printf("\n First(%c) = { ", c);
        store_first[point1][point2++] = c;
```

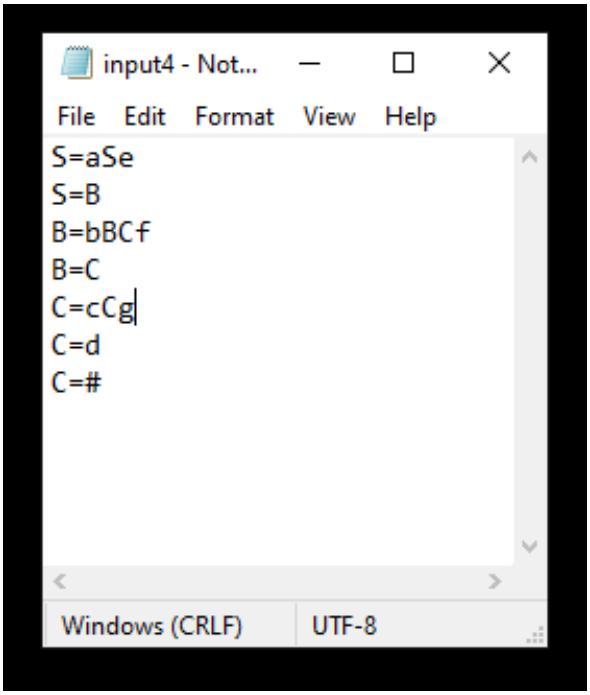
```
    for (i = 0 + jm; i < n; i++)
    {
        int lark = 0, chk = 0;

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

            if (first[i] == store_first[point1][lark])
            {
                chk = 1;
                break;
            }
        }
        if (chk == 0)
        {
            printf("%c, ", first[i]);
            store_first[point1][point2++] = first[i];
        }
    }
    printf("}\n");
    jm = n;
    point1++;
}
printf("\n\n\n");

char donee[cnt];
ptr = -1;


//calculate follow set
for (k = 0; k < cnt; k++)
{
    for (kay = 0; kay < 100; kay++)
    {
        store_follow[k][kay] = '!';
    }
}
point1 = 0;
```

```
    int land = 0;
    for (e = 0; e < cnt; e++)
    {
        ck = production[e][0];
        point2 = 0;
        x = 0;

        for (kay = 0; kay <= ptr; kay++)    // Checking already been
calculated
            if (ck == donee[kay])
                x = 1;

        if (x == 1)
            continue;
        land += 1;

        //function call
        follow(ck);

        ptr += 1;
        donee[ptr] = ck;
        printf(" Follow(%c) = { ", ck);
        store_follow[point1][point2++] = ck;

        //printing follow
        for (i = 0 + km; i < m; i++)
        {
            int lark = 0, chk = 0;
            for (lark = 0; lark < point2; lark++)
            {
                if (f[i] == store_follow[point1][lark])
                {
                    chk = 1;
                    break;
                }
            }
            if (chk == 0)
            {
                printf("%c, ", f[i]);
                store_follow[point1][point2++] = f[i];
```

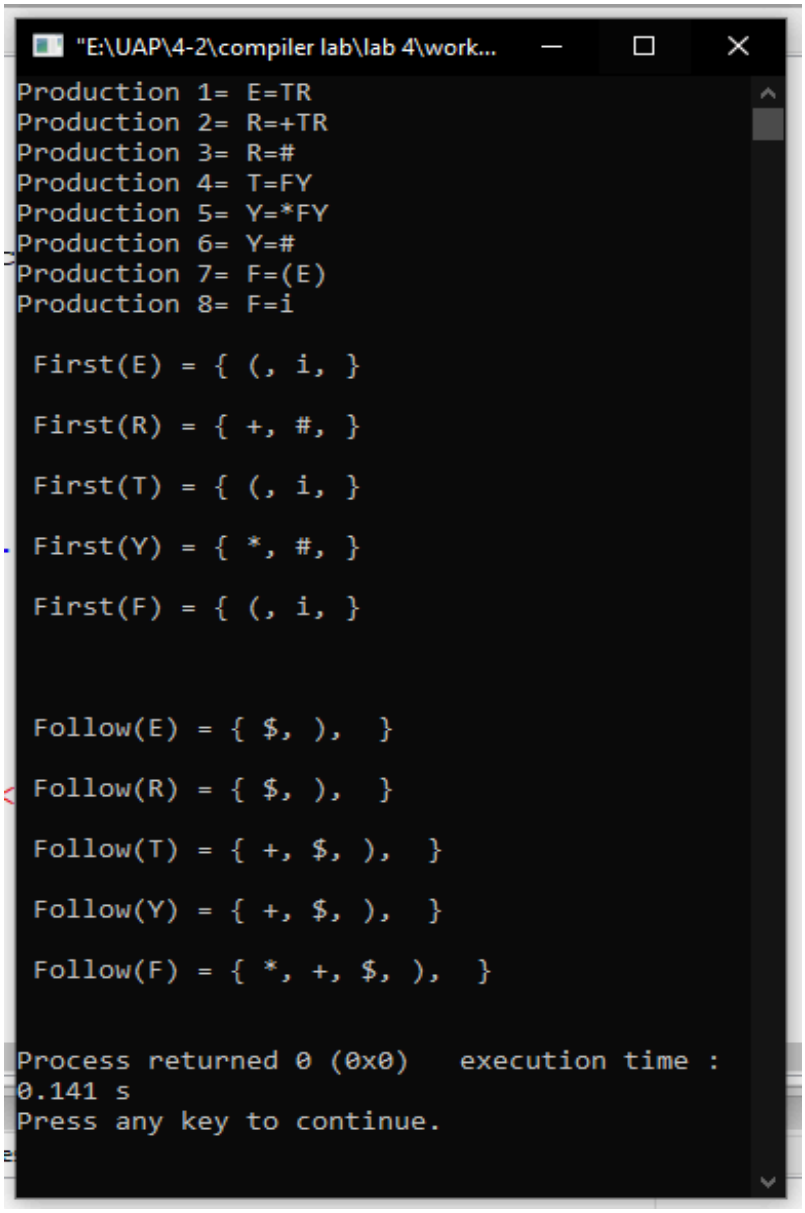
```
            }
        }
        printf(" }\n\n");
        km = m;
        point1++;
    }
}
```

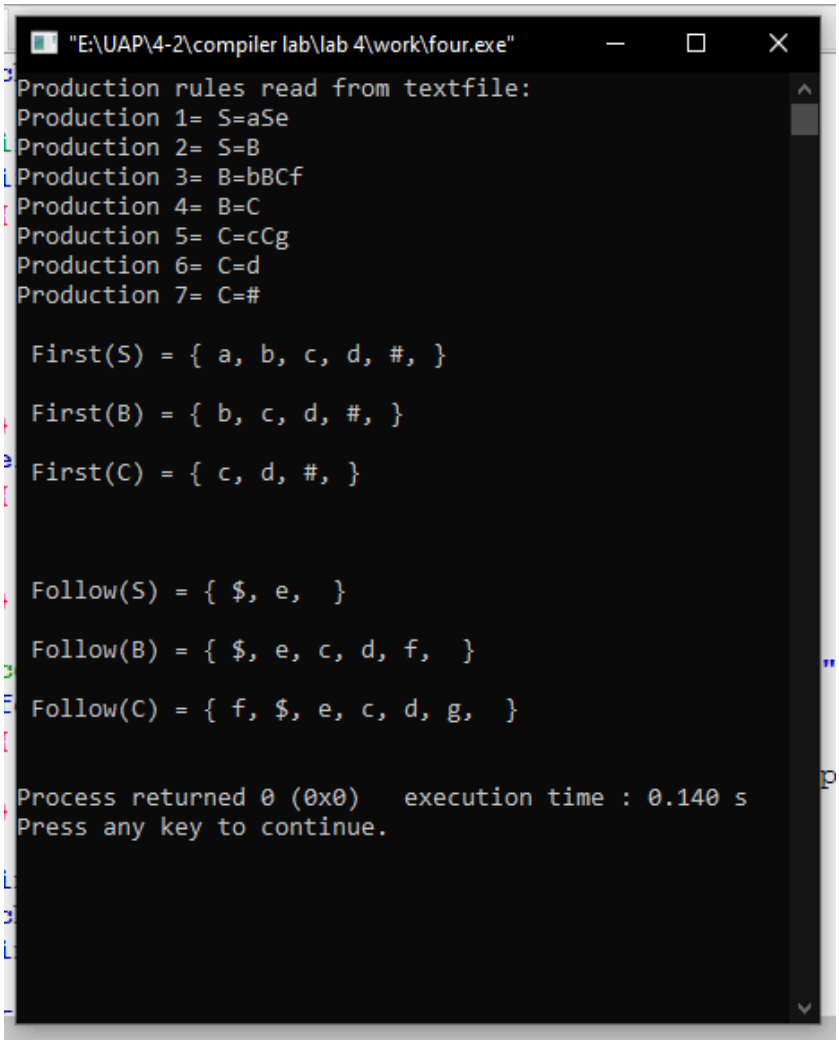
Sample Input:





Observed Output




"E:\UAP\4-2\compiler lab\lab 4\work\four.exe"
Production rules read from textfile:
Production 1= S=aSe
Production 2= S=B
Production 3= B=bBCf
Production 4= B=C
Production 5= C=cCg
Production 6= C=d
Production 7= C=#
First(S) = { a, b, c, d, #, }
First(B) = { b, c, d, #, }
First(C) = { c, d, #, }
Follow(S) = { $, e, }
Follow(B) = { $, e, c, d, f, }
Follow(C) = { f, $, e, c, d, g, }
Process returned 0 (0x0) execution time : 0.140 s
Press any key to continue.