COMPILER DESIGN LAB (CSL5404)

Name: Lakhan Kumawat

Roll: 1906055

Program: B.Tech CSE (5th Sem JUL-DEC 2021)

Assignment - 7

Name : Lakhan Kumawat Roll No : 1906055 Course : CSL5404

1 . Consider a given CFG.

E → E + T / T
T → T * F/ F
F → id

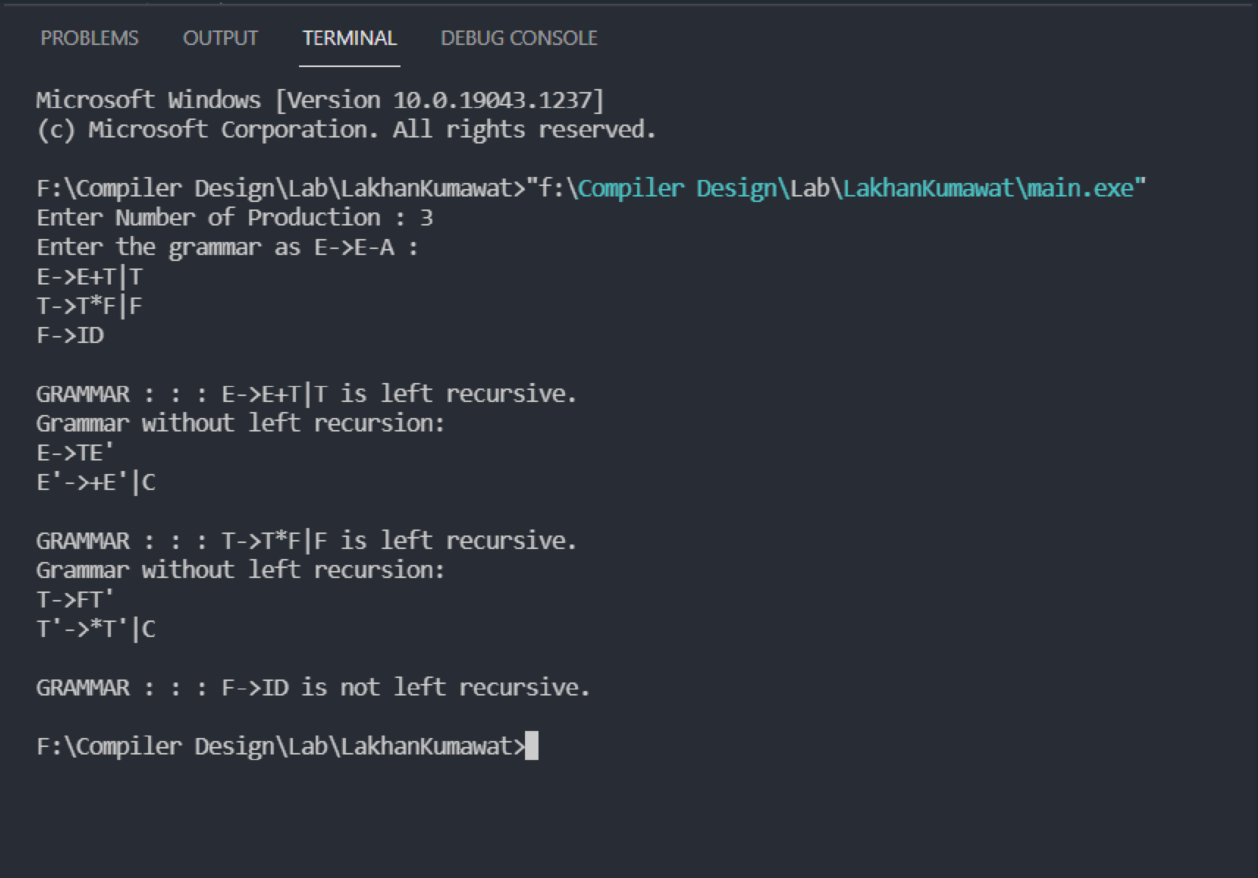
A> And write C program for performing following two tasks
Eliminate left recursion.

B> Find First and Follow sets of a
Grammar.

Code A:

```
#include <stdio.h>
#include<string.h>
#define SIZE 10
int main()
{
   char non_terminal;
       char beta,alpha;
       int num;
       char production[10][SIZE];
       int index=3; /* starting of the string following "-
>" */
       printf("Enter Number of Production : ");
       scanf("%d",&num);
       printf("Enter the grammar as E->E-A :\n");
       for(int i=0;i<num;i++){
            scanf("%s",production[i]);
       }
       for(int i=0;i<num;i++){
            printf("\nGRAMMAR : : : %s",production[i]);
            non_terminal=production[i][0];
            if(non_terminal==production[i][index]) {
                 alpha=production[i][index+1];
                 printf(" is left recursive.\n");
                 while(production[i][index]!=0 && production[i]
[index]!='|')
                      index++;

                 if(production[i][index]!=0) {
                      beta=production[i][index+1];
                      printf("Grammar without left recursion:\n
");
                      printf("%c-
>%c%c\'",non_terminal,beta,non_terminal);
                      printf("\n%c\'-
>%c%c\'|C\n",non_terminal,alpha,non_terminal);
                 }
                 else
                      printf(" can't be reduced\n");
            }
            else
                 printf(" is not left recursive.\n");
            index=3;
       }
       return 0;
  }
```

Name : Lakhan Kumawat Roll No : 1906055 Course : CSL5404



```
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

F:\Compiler Design\Lab\LakhanKumawat>"f:\Compiler Design\Lab\LakhanKumawat\main.exe"
Enter Number of Production : 3
Enter the grammar as E->E-A :
E->E+T|T
T->T*F|F
F->ID

GRAMMAR : : : E->E+T|T is left recursive.
Grammar without left recursion:
E->TE'
E'->+E'|C

GRAMMAR : : : T->T*F|F is left recursive.
Grammar without left recursion:
T->FT'
T'->*T'|C

GRAMMAR : : : F->ID is not left recursive.

F:\Compiler Design\Lab\LakhanKumawat>
```

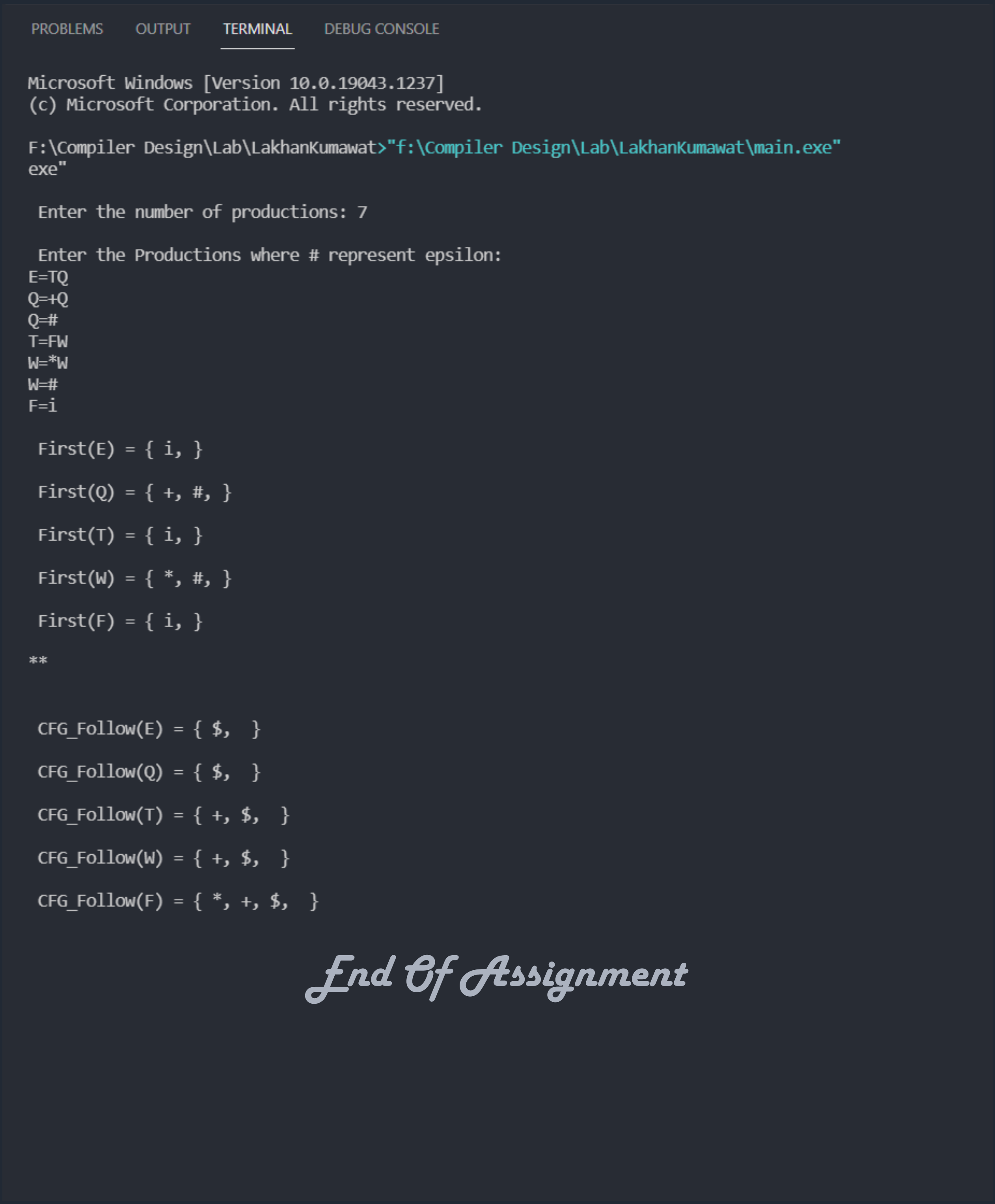
2 > Program Code

```c
#include <stdio.h>
#include <ctype.h>
#include <string.h>
// Functions to calculate CFG_Follow
void followFirst(char, int, int);
void CFG_Follow(char c);
// Function to calculate First
void Searching_First(char, int, int);
int count, n = 0;
// Stores the final result of the First Sets
char FirstCalculation[10][100];
// Stores the final result of the CFG_Follow Sets
char calc_CFG_Follow[10][100];
```

Name : Lakhan Kumawat Roll No : 1906055 Course : CSL5404

```
int m = 0;
// Stores the production rules
char production[10][10];
char f[10], first[10];
int k;
char ck;
int e;
int main(int argc, char **argv)
{
    int jm = 0;
    int km = 0;
    int i, choice;
    char c, ch;
    int num;
    printf("\n Enter the number of productions: ");
    scanf("%d", &num);
    printf("\n Enter the Productions where # represent epsilon:
 \n");
    for (int i = 0; i < num; i++)
    {
        scanf("%s", &production[i]);
    }
    count = num;
    int kay;
    char done[count];
    int ptr = -1;

    for (k = 0; k < count; k++)
    {
        for (kay = 0; kay < 100; kay++)
        {
            FirstCalculation[k][kay] = '!';
        }
    }
    int point1 = 0, point2, xxx;
    for (k = 0; k < count; k++)
    {
        c = production[k][0];
        point2 = 0;
        xxx = 0;

        for (kay = 0; kay <= ptr; kay++)
            if (c == done[kay])
                xxx = 1;
        if (xxx == 1)
            continue;

        Searching_First(c, 0, 0);
        ptr += 1;

        done[ptr] = c;
        printf("\n First(%c) = { ", c);
        FirstCalculation[point1][point2++] = c;

        for (i = 0 + jm; i < n; i++)
        {
            int lark = 0, chk = 0;
            for (lark = 0; lark < point2; lark++)
            {
                if (first[i] == FirstCalculation[point1][lark])
                {
                    chk = 1;
                    break;
                }
            }
            if (chk == 0)
            {
                printf("%c, ", first[i]);
                FirstCalculation[point1][point2++] = first[i];
            }
        }
        printf("}\n");
        jm = n;
        point1++;
    }
    printf("\n");
    printf("**\n\n\n");
    char donee[count];
    ptr = -1;

    for (k = 0; k < count; k++)
    {
        for (kay = 0; kay < 100; kay++)
        {
            calc_CFG_Follow[k][kay] = '!';
        }
    }
    point1 = 0;
    int land = 0;
    for (e = 0; e < count; e++)
    {
        ck = production[e][0];
        point2 = 0;
        xxx = 0;

        for (kay = 0; kay <= ptr; kay++)
            if (ck == donee[kay])
                xxx = 1;
        if (xxx == 1)
            continue;
        land += 1;

        CFG_Follow(ck);
        ptr += 1;

        donee[ptr] = ck;
        printf(" CFG_Follow(%c) = { ", ck);
        calc_CFG_Follow[point1][point2++] = ck;
        for (i = 0 + km; i < m; i++)
        {
            int lark = 0, chk = 0;
            for (lark = 0; lark < point2; lark++)
            {
                if (f[i] == calc_CFG_Follow[point1][lark])
                {
                    chk = 1;
                    break;
                }
            }
            if (chk == 0)
            {
                printf("%c, ", f[i]);
                calc_CFG_Follow[point1][point2++] = f[i];
            }
        }
        printf(" }\n\n");
        km = m;
        point1++;
    }
}
void CFG_Follow(char c)
{
    int i, j;

    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')
                {

                    followFirst(production[i][j + 1], i, (j + 2
));
                }
                if (production[i][j + 1] == '\0' && c != produc
tion[i][0])
                {

                    CFG_Follow(production[i][0]);
                }
            }
        }
    }
}
void Searching_First(char c, int q1, int q2)
{
    int j;

    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))
                {
                    Searching_First(production[q1][q2], q1, (q2
 + 1));
                }
                else
                    first[n++] = '#';
            }
            else if (!isupper(production[j][2]))
            {
                first[n++] = production[j][2];
            }
            else
            {

                Searching_First(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 < count; i++)
        {
            if (FirstCalculation[i][0] == c)
                break;
        }

        while (FirstCalculation[i][j] != '!')
        {
            if (FirstCalculation[i][j] != '#')
            {
                f[m++] = FirstCalculation[i][j];
            }
            else
            {
                if (production[c1][c2] == '\0')
                {

                    CFG_Follow(production[c1][0]);
                }
                else
                {

                    followFirst(production[c1][c2], c1, c2 + 1)
;
                }
            }
            j++;
        }
    }
}
```

Name : Lakhan Kumawat Roll No : 1906055 Course : CSL5404



```
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

F:\Compiler Design\Lab\LakhanKumawat>"f:\Compiler Design\Lab\LakhanKumawat\main.exe"
exe"

 Enter the number of productions: 7

 Enter the Productions where # represent epsilon:
E=TQ
Q=+Q
Q=#
T=FW
W=*W
W=#
F=i

 First(E) = { i, }

 First(Q) = { +, #, }

 First(T) = { i, }

 First(W) = { *, #, }

 First(F) = { i, }

**


 CFG_Follow(E) = { $,  }

 CFG_Follow(Q) = { $,  }

 CFG_Follow(T) = { +, $,  }

 CFG_Follow(W) = { +, $,  }

 CFG_Follow(F) = { *, +, $,  }
```

End Of Assignment