CS6612 – Compiler Lab

Ex no : 3 **Name : Sreedhar V**

Date : 20.02.2021 **Reg no: 185001161**

Specification

Write a program in C to find whether the given grammar is Left Recursive or not. If it is found to be left recursive, convert the grammar in such a way that the left recursion is removed.

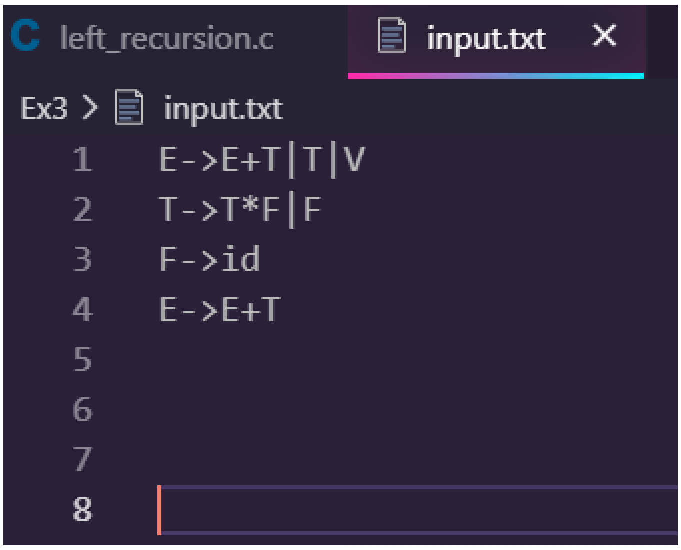
Code

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
void detect(char prod[][30],int n)
{
    for(int i=0;i<n;++i)
   {
        char p = prod[i][0];
        if(p==prod[i][3])
        {
            char *token = strtok(prod[i],"|");
            char alpha[10];
            int j=0;
            for(int itr=4;itr<strlen(token);itr++)
                alpha[j++]=token[itr];
            char beta[10][20];
            j=0;
            char buffer[20];
            while(token!=NULL)
            {
                strcpy(buffer,token);
                token=strtok(NULL,"|");
                if(token!=NULL)
                    strcpy(beta[j++],token);
                else
                    strcpy(beta[j++],buffer);
            }
            j--;
            if(!j)
            {
                printf("\n%c -> %c'",p,p);
                char alpha[10];
                int j=0;
                for(int itr=4;itr<strlen(prod[i]);itr++)
                    alpha[j++]=prod[i][itr];
                printf("\n%c'->%s %c' | (null)\n",p,alpha,p);
                continue;
            }
            printf("\n%c ->%s %c'",p,beta[0],p);
            for(int i=1;i<j;i++)
                printf("|%s %c'",beta[i],p);
            printf("\n%c'->%s %c' | (null)\n",p,alpha,p);
        }
        else
            printf("\n%s\n",prod[i]);
    }
}
int main()
{
    int n=0;
    char prod[20][30];

    printf("\n\tLeft Recursion_Elimation\n");
    int i=0;
    FILE *file = fopen("input.txt","r");
    char c;
    printf("\nGiven grammar\n");
    while(fscanf(file,"%c",&c)==1)
    {
        if(c=='\n')
        {
            prod[n][i]='\0';
            printf("\n%s\n",prod[n]);
            n++;
            i=0;
        }
        else
        {
            prod[n][i]=c;
            i++;
        }
    }
    printf("The set of productions in grammer after left recursion:\n")
;
    detect(prod,n);
```

```c
    printf("\n");
    return 0;
}
```

(Sample input file)

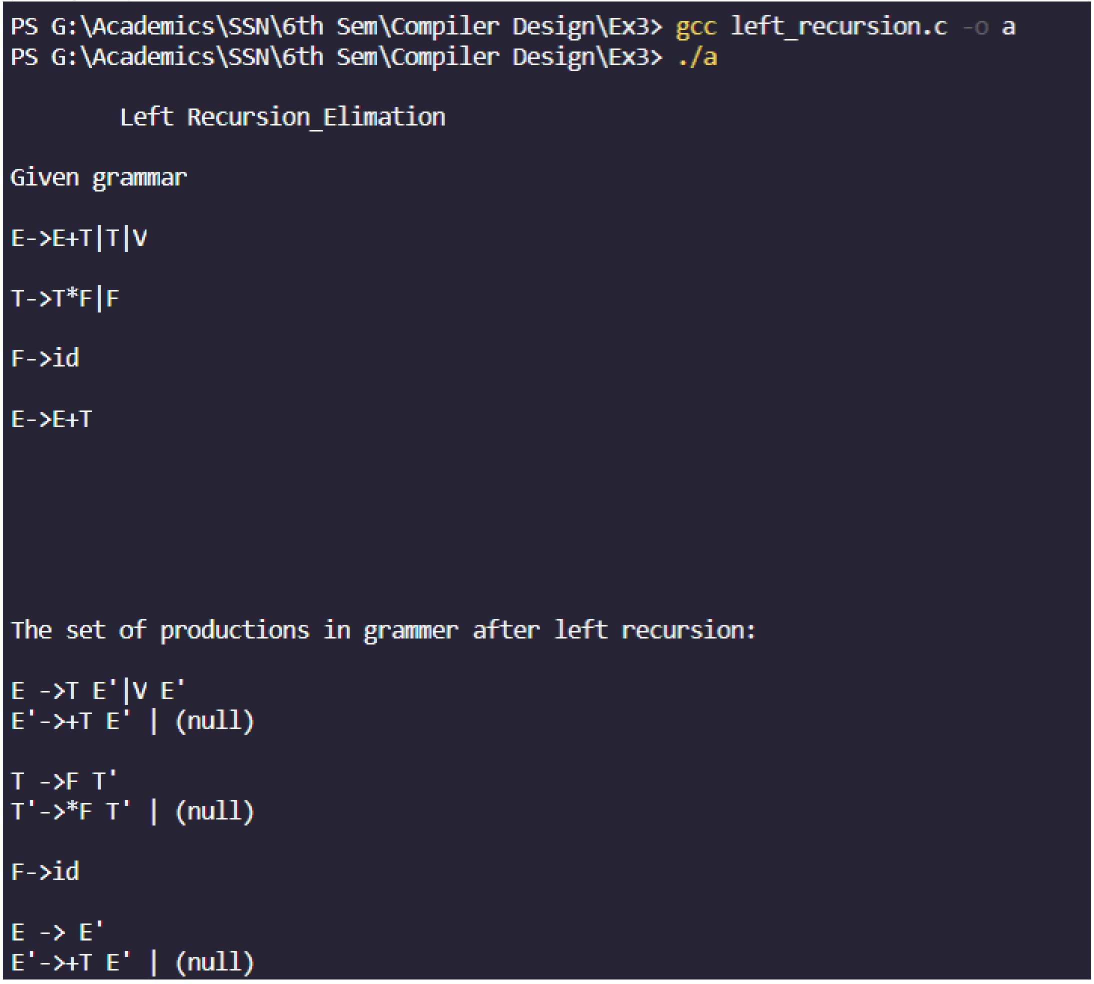


```
E->E+T|T|V
T->T*F|F
F->id
E->E+T



```

(Output)



```
PS G:\Academics\SSN\6th Sem\Compiler Design\Ex3> gcc left_recursion.c -o a
PS G:\Academics\SSN\6th Sem\Compiler Design\Ex3> ./a

        Left Recursion_Elimation

Given grammar

E->E+T|T|V

T->T*F|F

F->id

E->E+T





The set of productions in grammer after left recursion:

E ->T E'|V E'
E'->+T E' | (null)

T ->F T'
T'->*F T' | (null)

F->id

E -> E'
E'->+T E' | (null)
```

Learning Outcome:

- I’ve learnt how to identify the left recursion in the production of the grammar and construct a new grammar with removing such productions.
- I’ve learnt how to implement the same using C code which identifies whether the grammar is left recursive or not and converts the grammar in a such a way that the left recursion is removed.
- I’ve learnt the Elimination of Immediate Left Recursion using the rule if the production is in the form A->Aα | β, then A-> βA’,A’->Ɛ | αA’.