UCS 1602 - Compiler Design

Exercise 3 - Implementation of Left Recursion Elimination using LEX tool

Name:	Mahesh Bharadwaj K
Reg No:	185001089
Semester:	VI
Date:	February 16, 2021

Aim:

To implement Left Recursion Elimination using C program

Program

Header file

```
typedef struct
{
    char left[10];
    char right[128];
} Production;

Production *eliminate_left_recursion(Production, int *);
void getProduction(Production *);
void putProduction(const Production);

void getProduction(Production *p)
{
    bzero(p, sizeof(Production));
    scanf("%s --> %s", p->left, p->right);
}

void putProduction(const Production p)
{
    printf("%s --> %s\n", p.left, p.right);
}

Production *eliminate_left_recursion(Production p, int *count)
{
    Production *p_list = calloc(2, sizeof(Production));
    strcpy(p_list[0].left, p.left);

    char buffer[128], p_right_copy[128];
    strcpy(p_right_copy, p.right);
    char *token = strtok(p_right_copy, "|");
    int lr_detected = 0;
    while (token != NULL)
    {
        if (token[0] == p.left[0])
        {
            lr_detected = 1;
            break;
        }
```

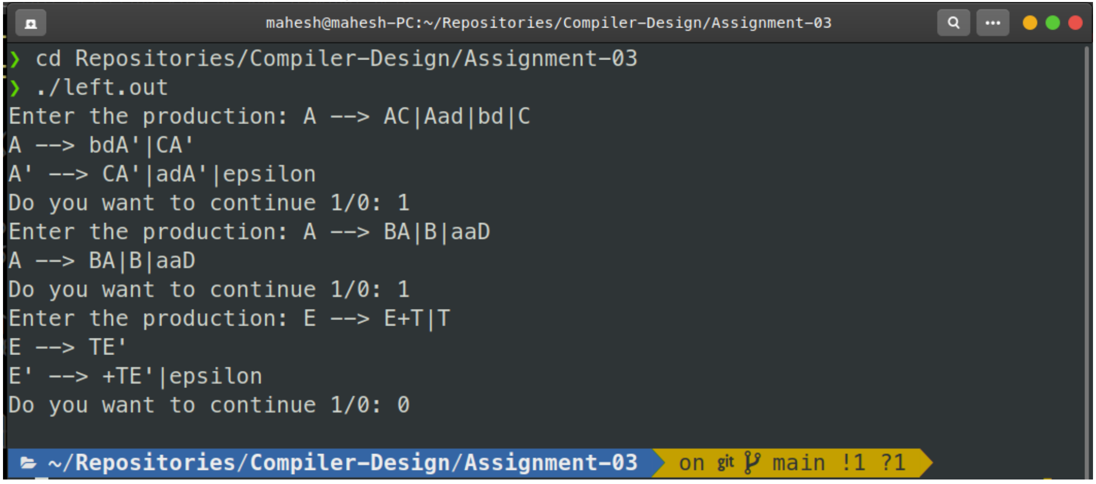
```c
        token = strtok(NULL, "|");
    }

    if (!lr_detected)
    {
        strcpy(p_list[0].left, p.left);
        strcpy(p_list[0].right, p.right);
        (*count) = 1;
        return p_list;
    }
    strcat(p_list[1].left, p.left);
    strcat(p_list[1].left, "'");
    strcpy(p_right_copy, p.right);
    token = strtok(p_right_copy, "|");
    while (token != NULL)
    {
        if (token[0] == p.left[0])
        {
            lr_detected = 1;
            strcat(p_list[1].right, &token[1]);
            buffer[0] = p.left[0];
            buffer[1] = '\'';
            buffer[2] = '|';
            buffer[3] = 0;
            strcat(p_list[1].right, buffer);
        }
        else
        {
            strcat(p_list[0].right, token);
            buffer[0] = p.left[0];
            buffer[1] = '\'';
            buffer[2] = '|';
            buffer[3] = 0;
            strcat(p_list[0].right, buffer);
        }
        token = strtok(NULL, "|");
    }
    p_list[0].right[strlen(p_list[0].right) - 1] = 0;
    strcat(p_list[1].right, "epsilon");
    (*count) = 2;
    return p_list;
}
```

Main Program

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

#include "production.h"

int main()
{
    char buffer[256] = {0};
    int count = 0;
    int opt;
    Production p = {0}, *p_list = NULL;
    do
    {
        printf("Enter the production: ");
        getProduction(&p);

        p_list = eliminate_left_recursion(p, &count);
        for (int i = 0; i < count; i++)
            putProduction(p_list[i]);
        free(p_list);

        printf("Do you want to continue 1/0: ");
        scanf("%d", &opt);
    } while (opt);
}
```

Output

Figure 1: **Sample Input and Output**



mahesh@mahesh-PC:~/Repositories/Compiler-Design/Assignment-03

```
❯ cd Repositories/Compiler-Design/Assignment-03
❯ ./left.out
Enter the production: A --> AC|Aad|bd|C
A --> bdA'|CA'
A' --> CA'|adA'|epsilon
Do you want to continue 1/0: 1
Enter the production: A --> BA|B|aaD
A --> BA|B|aaD
Do you want to continue 1/0: 1
Enter the production: E --> E+T|T
E --> TE'
E' --> +TE'|epsilon
Do you want to continue 1/0: 0

 ~/Repositories/Compiler-Design/Assignment-03  on git main !1 ?1
```

Learning Outcomes

1. We learn to identify left recursion
2. We learn to remove left recursion
3. We learn to write C program to remove left recursion, if present from the productions given