**Practical 4**

**Compiler Construction**

2CS701

**Mistry Unnat**

20BCE515



Department of Computer Science and Engineering

Institute of Technology

Nirma University

Ahmedabad

**Aim :**

**To Implement Left Recursion derivation removal algorithm :**

**Eliminate direct and indirect Left recursion from given grammar for LL(1) parser.**

**Input :**

E->EA|A

A->AT|a

**C code :**

  #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\'|E\n",non\_terminal,alpha,non\_terminal);

                }

                else

                     printf(" can't be reduced\n");

           }

           else

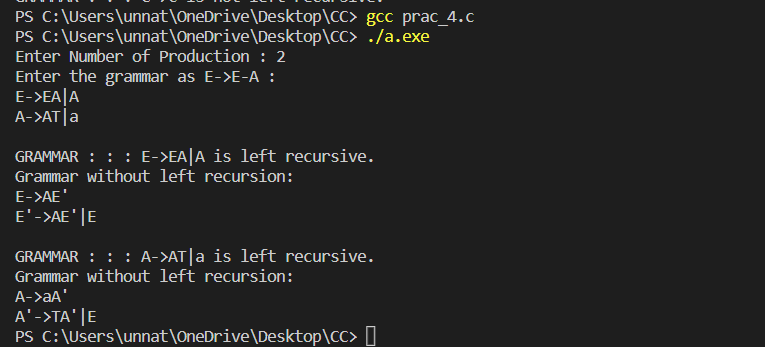
                printf(" is not left recursive.\n");

           index=3;

      }

 }

**Output :**



**Conclusion :**

From this practical, we learnt about elimination of direct and indirect left recursion.