Exp. No. 9

Implement a C program to eliminate left recursion from a given CFG.

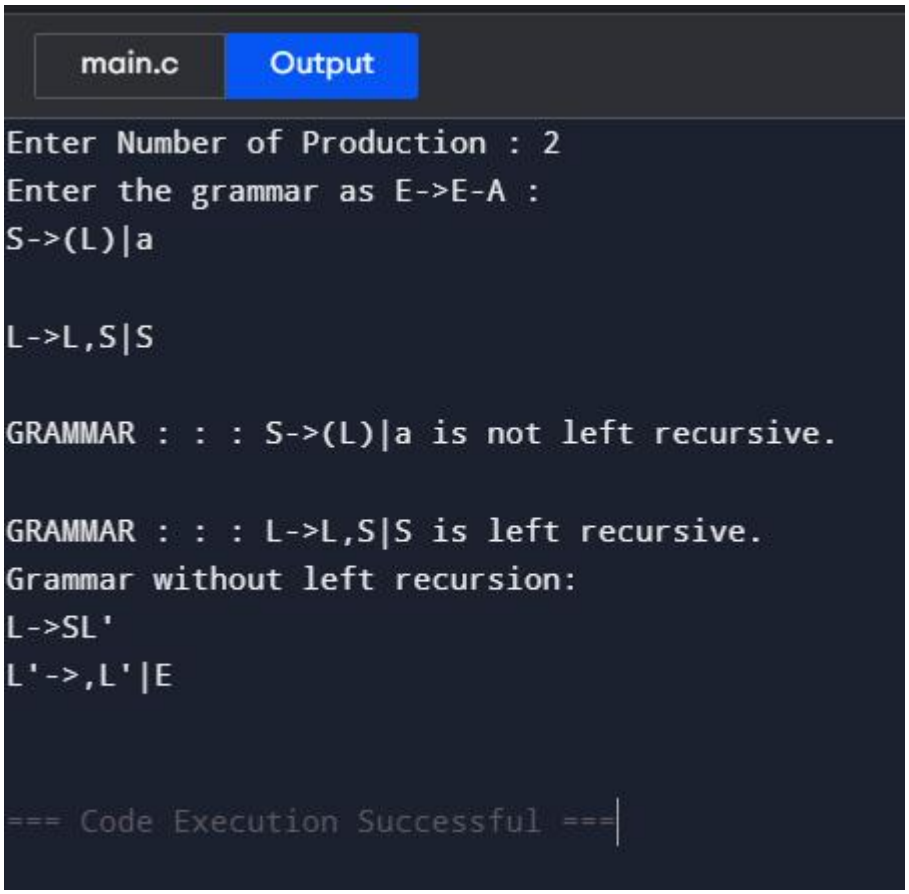
S → (L) / a

L → L , S / S

Program:

```
#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;
}
```

}


main.c
Output
Enter Number of Production : 2
Enter the grammar as E->E-A :
S->(L)|a
L->L,S|S
GRAMMAR : : : S->(L)|a is not left recursive.
GRAMMAR : : : L->L,S|S is left recursive.
Grammar without left recursion:
L->SL'
L'->,L'|E
=== Code Execution Successful ===