# REMOVING LEFT RECURSSION

```
def check_left_recursion(A,prod):
    for x in prod:
        if len(x)>1 and x[0] == A:
            return True
    return False



n=input("\ninput number of non terminals: ")
n=int(n)
NT =[]
productions = []

NewNT = []
NewProd = []



for i in range(n):
    NT.append(input(f"\nenter non terminal {i+1}: "))
print("\n use @ for epsilon\n")
for x in NT:
    K = input(f"\nenter productions for {x} seperated by pipe(|): ")
    productions.append(K.split("|"))

for i in zip(NT,productions):
```

```
if(check_left_recursion(i[0],i[1])):
    alpha = []
    beta =[]
    print(f"\nfor {i[0]}--> {i[1]} Left recursion occurs")
    for prod in i[1]:
        if len(prod) > 1 and i[0] == prod[0]:
            alpha.append(prod[1:])
        else:
            beta.append(prod)
    NewNT.append(i[0])
    prod1 =[]
    for x in beta:
        if(x!='@'):# @ is epsilon
            prod1.append(f"{x}{i[0]}\'")
        else:
            prod1.append(f"{i[0]}\'")
    NewProd.append(prod1);
    NewNT.append(f"{i[0]}\'")
    prod2 = []
    for x in alpha:
        prod2.append(f"{x}{i[0]}\'")
    prod2.append("@")
    NewProd.append(prod2);

else:
    NewNT.append(i[0])
    NewProd.append(i[1])
```

```
for x,p in zip(NewNT,NewProd):
    string = ""SS
```

**OUTPUT**

OUTPUT 1

```
input number of non terminals: 2
enter non terminal 1: S
enter non terminal 2: A
 use @ for epsilon

enter productions for S seperated by pipe(|): ab|A
enter productions for A seperated by pipe(|): Ad|h|@
for A--> ['Ad', 'h', '@'] Left recursion occurs
S --> ab|A
A --> hA'|A'
A' --> dA'|@
```

OUTPUT 2

```
input number of non terminals: 1
enter non terminal 1: S
 use @ for epsilon

enter productions for S seperated by pipe(|): aSb|bSa|SS|@
for S--> ['aSb', 'bSa', 'SS', '@'] Left recursion occurs
S --> aSbS'|bSaS'|S'
S' --> SS'|@
```

OUTOUT 3

```
input number of non terminals: 3
enter non terminal 1: A
enter non terminal 2: B
enter non terminal 3: C
 use @ for epsilon

enter productions for A seperated by pipe(|): Ad|Ae|aB|aC
enter productions for B seperated by pipe(|): Bbc|f
enter productions for C seperated by pipe(|): g
for A--> ['Ad', 'Ae', 'aB', 'aC'] Left recursion occurs
for B--> ['Bbc', 'f'] Left recursion occurs
A --> aBA'|aCA'
A' --> dA'|eA'|@
B --> fB'
B' --> bcB'|@
C --> g
```