Experiment - 5 WAP to remove left recursion from a grammar

Name: **Rollno :**
Class : TE.CO **Batch :**

#Source Code

```
__author__ = 'Shadab Shaikh'
__title__ = 'Finding & resolving left recursion from a grammar'
__date__ = '26-02-2019'
__version__ = '1.0'

print('Author          : ' + __author__)
print('Title           : ' + __title__)
print('Date            : ' + __date__)
print('Version         : ' + __version__)

grammararr=[]       #stores the grammar maintaining the index
alpharest=[]        #stores the alpha and rest values
beta=[]             #stores beta
inputs=""
newprod=""          #to display new production
while(inputs!='no'):
        grammar = input("\nEnter the grammar left should be variable following with ->
format eg: S->a\n")
        grammar=grammar.replace(" ","")          #replacing whitespaces with none
        if(grammar[0].islower()):
                grammar[0].upper()               #making left most as variable
        grammararr.append(grammar)               #storing into list
        inputs = input("\nPress no to stop writing productions or write anything to continue")
                                                 #asking for the continuity of grammar



def findalpharest(grammararr,k):
        """function to find alpha and rest values"""
        if(grammararr[k][0]==grammararr[k][3]):
                alpharest.append(grammararr[k][0]+grammararr[k][4:])
        k+=1
        if(k<len(grammararr)):
                findalpharest(grammararr,k)


findalpharest(grammararr,0)                      #calling findalpharest function

def findbeta(grammararr,alpharest,l,m):
```

```
        """function to find beta for corresponding alpha values"""
        if(grammararr[l][0]==alpharest[m][0]):
                if(grammararr[l][0]!=grammararr[l][3]):
                        beta.append(grammararr[l][0]+grammararr[l][3])
        l+=1
        if(l<len(grammararr)):
                findbeta(grammararr,alpharest,l,m)
        else:
                l=0
                m+=1
                if(m<len(alpharest)):
                        findbeta(grammararr,alpharest,l,m)

findbeta(grammararr,alpharest,0,0)    #calling findbeta function

def formnewprod(grammararr,alpharest,beta,i,j):
        #function to print new production
        print(alpharest[i][0]+"->"+beta[j][1]+alpharest[i][0]+"'")
        print(alpharest[i][0]+"'"+"->"+alpharest[i][1]+alpharest[i][0]+"'")
        print(alpharest[i][0]+"'"+"#")
        i+=1
        j+=1
        if(i<len(alpharest) and j<len(beta)):
                formnewprod(grammararr,alpharest,beta,i,j)


formnewprod(grammararr,alpharest,beta,0,0)          #calling formnewprod function

#print(*alpharest, sep='\n')
#print(*beta, sep='\n')
```

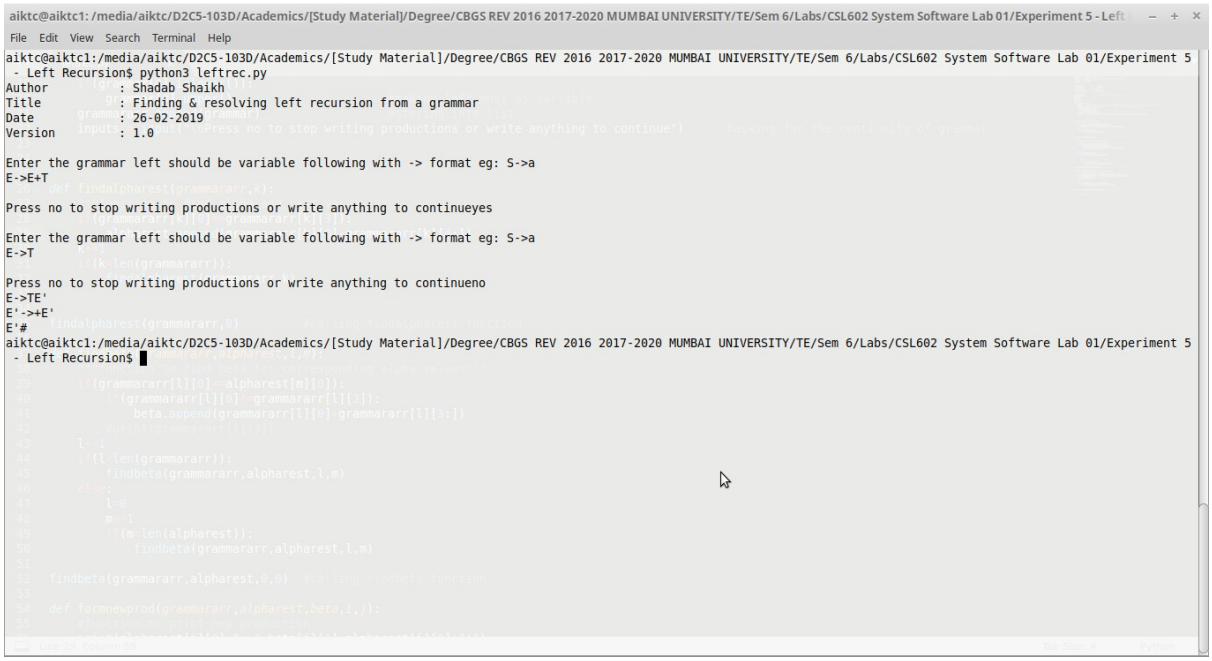
#Sample input

1. **E->E+T/T**

2. **S->(L)/x**
   L-Ls/s

#Output

1. Output for the 1st sample input



2. Output for the 2nd sample input