Write a program to check and remove left recursion.[direct/indirect]

Intro:

- A production of grammar is said to have **left recursion** if the leftmost variable of its RHS is same as variable of its LHS.

Code:

print("Enter number of productions")

num = int(input())

productions = []

print("Enter as E->EA|a \n")

while num > 0:

    pr = input().replace(" ", "")

    productions.append(pr)

    num -= 1

print()

# print(productions)

for prod in productions:

    if prod.count('|') == 1:

        if prod[0] == prod[3]:

            print("{} is left recursive".format(prod))

            alpha = prod[4]

            bitaIndex = prod.index('|') + 1

            bita = prod[bitaIndex:]

            print("After removing recursion ::")

            print(f"{prod[0]} -> {bita} {prod[0]}'")

            print(f"{prod[0]}' -> {alpha} {prod[0]}' | ∅")

            print("-------------------------------------")

    elif prod.count('|') > 1:

        if prod[0] == prod[3]:

            # print("{} is left recursive".format(prod))

            prod = prod.split('->')

            nonTerminals = prod[1:][0].split('|')

            alpha = f"{prod[0]}' -> "

            bita = f"{prod[0]} -> "

            for i in nonTerminals:

                if len(i)>1:

                    if prod[0] == i[0]:

                        alpha += f"{i[1:]}{prod[0]}' | "

                    elif prod[0] not in i:

                        bita += f"{i}{prod[0]}' | "

            print(bita[:-2])

            print(alpha + 'None')

            print("-------------------------------------")

Output:

