**KATHMANDU UNIVERSITY**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**Lab Work 3**

**COMP 316**

**SUBMITTED BY: SUBMITTED TO:**

Arun Regmi Sushil Nepal

Roll No: 66 DOCSE

Computer Science (3rd Year/ 1st Sem)

Date of Submission: February 12, 2020.

# 

**Problem**

**Q.** Write a program to eliminate left recursion in the given grammar.

**Task**

* First Check Whether the grammar is left recursive or not
* If left recursive
  + Apply Elimination Method and Display left recursion free Grammar
* Else
  + Print Grammar is not left recursive

**Key Features**

The program “lab3.py” compiles a given grammar structure given by the user in the program and eliminates the left recursion from the grammar. The compiled grammar is first checked to find out whether the grammar structure is left recursive grammar or not and carries out the work accordingly.

**Code:**

grammar\_s = input("Enter a Grammar: ") # "A->bba|abb|aab" #"B-> bbA|Bab"

# grammar\_s = "A->bba|abb|Aab" "S->aaS|bba" "G->aaS|Gas"

# print(grammar\_string)

a = grammar\_s[0]

allalphas = []

allbetas = []

splits = grammar\_s[3:].split('|')

# print(splitted)

for s in splits:

if a == s[0]:

allalphas.append(s[1:])

else:

allbetas.append(s)

betas\_new = [beta+a+"'" for beta in allbetas]

alphas\_new = [alpha+a+"'" for alpha in allalphas]

if len(allalphas)==0:

print("Not A Left Recursive Grammar")

else:

print("A Left Recursive Grammar")

print("Grammar: ")

print(f"{a}->{'|'.join(betas\_new)}")

print(f"{a}'->{'|'.join(alphas\_new)}|E")

**Description:**

The user inputs grammar which is stored in the “grammar\_s”. We split the splitted grammar into “allbetas” and “allalphas” array.

If length of “allalphas” is 0 then the grammar is not left recursive otherwise the grammar is left recursive.

We then join betas\_new and alphas\_new to form new grammar.

**Output:**

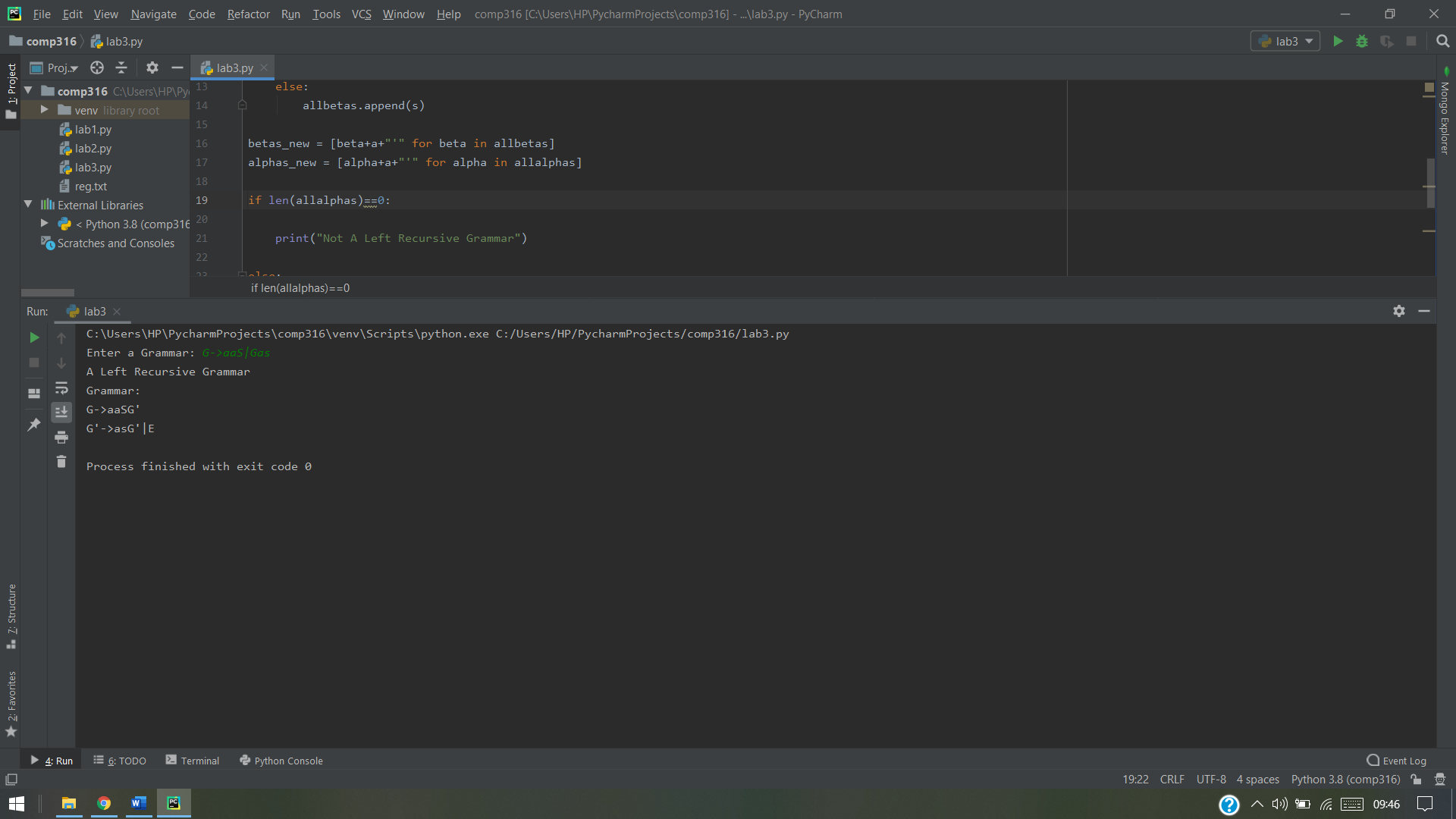


Image 1: Output if the grammar is left recursive

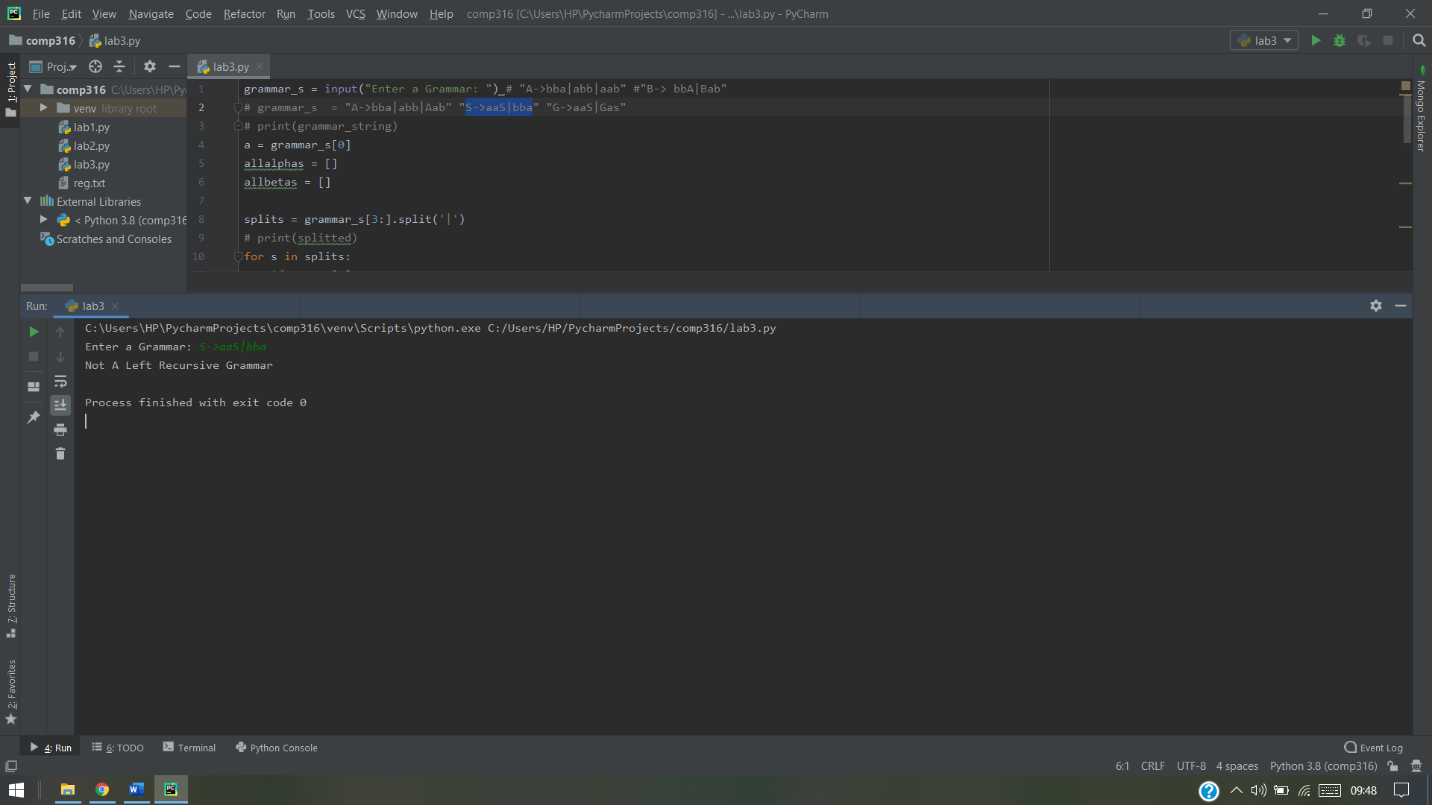


Image 2: Output if the grammar is not left recursive