**KATHMANDU UNIVERSITY**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**Lab Work 3**

**COMP 316**

**BY:**

Shubh Raj Dhital

Roll No : 70

Group : Computer Science

Department of Computer Science and Engineering

**Date of Submission: February 12, 202**

Problem :

Write a program to eliminate left recursion from 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 checks whether the provided grammar is left recursive or not .

If the grammar is left recursive, we apply Elimination Method and Display left recursion free Grammar else Print Grammar is not left recursive.

Code  :

lab3.py

grammar\_s = input("Enter a Grammar: ") #this input add  "R->wwr|rww|Rrw"

# grammar\_s  = "R->wwr|rww|Rrw"

# 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 :

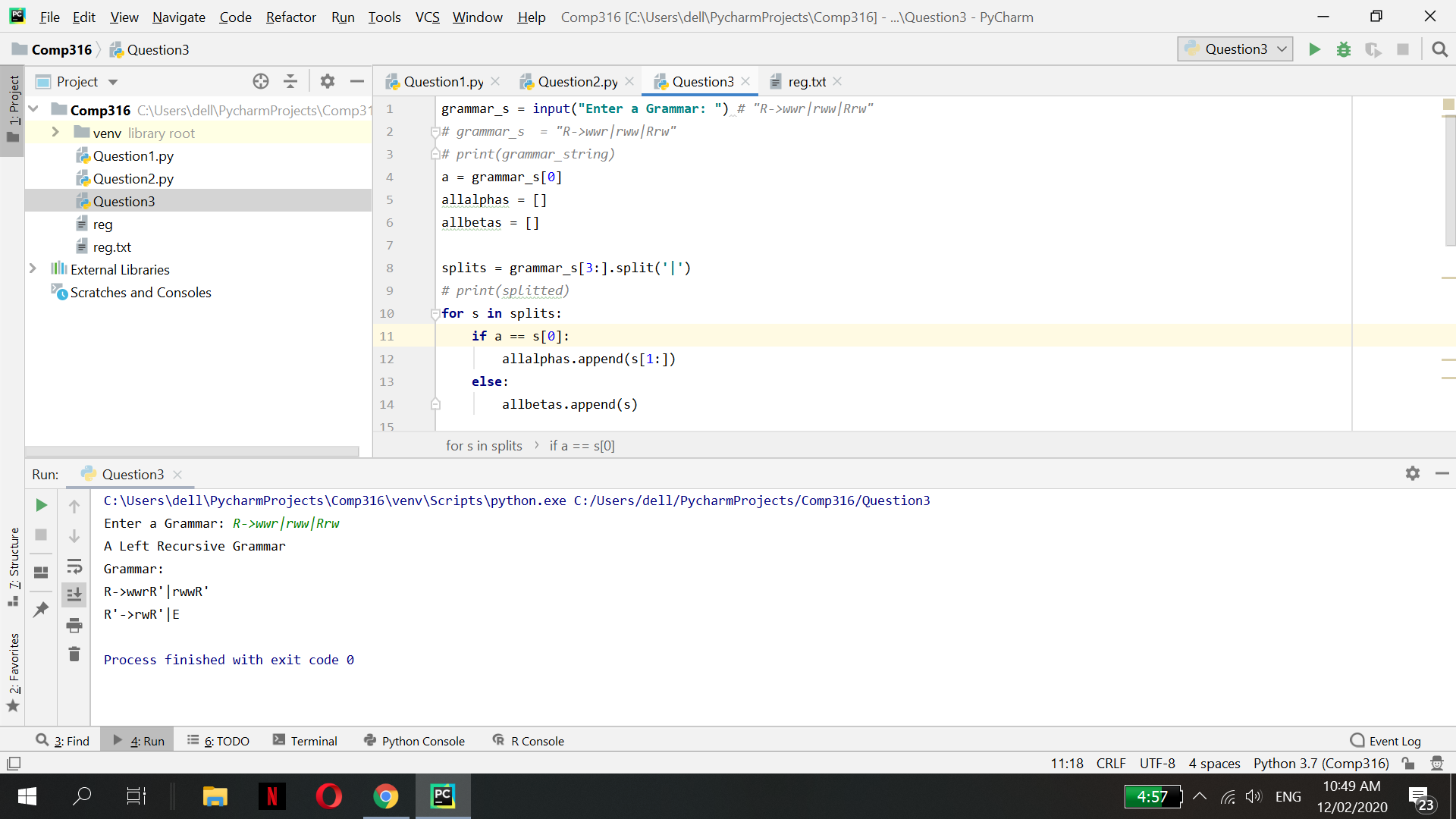


Figure: Output for left recursive grammar