University of Asia Pacific

Department of Computer Science & Engineering

Compiler Design Lab

CSE 430

Submitted to:

Baivab Das
Lecturer
CSE, University of Asia Pacific

Submitted by:

Asma Sultana
20101084
Section: B1

Lab 3

Problem

Elimination of Left Recursion in a grammar.

Description of the problem

There is a production in the form of:

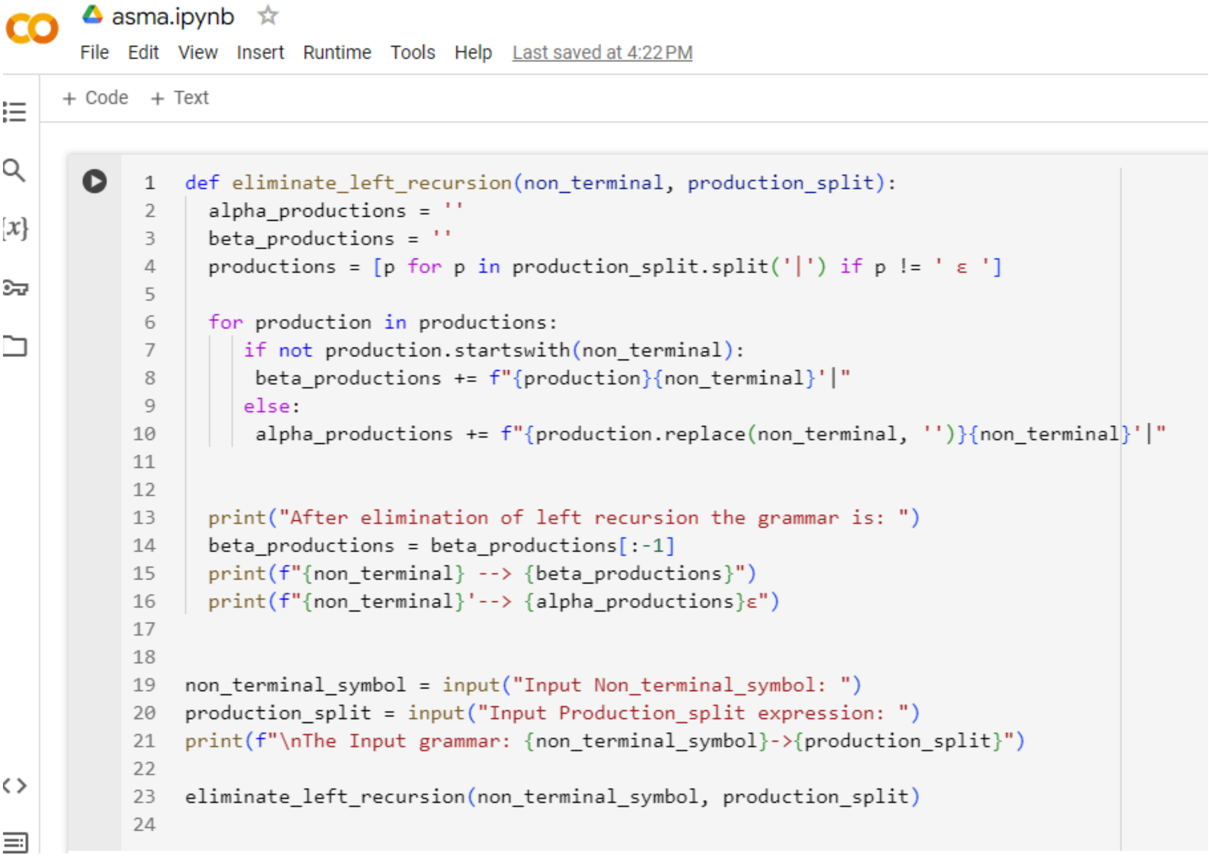
A →Aα

Now, we use the elimination left recursion algorithm to eliminate left recursion from a grammar which is given as an input in the console. Left recursion to consider

Now I generate the problem using this format-

A → βA’
A’ → αA’ | ε

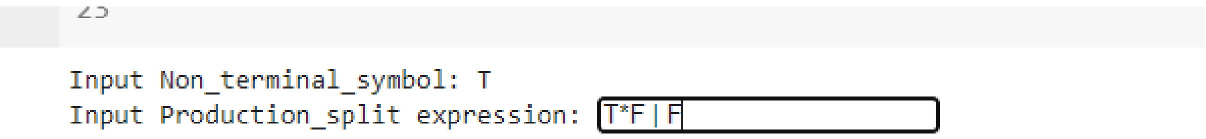
Code



```
def eliminate_left_recursion(non_terminal, production_split):
  alpha_productions = ''
  beta_productions = ''
  productions = [p for p in production_split.split('|') if p != ' ε ']

  for production in productions:
     if not production.startswith(non_terminal):
      beta_productions += f"{production}{non_terminal}'|"
     else:
      alpha_productions += f"{production.replace(non_terminal, '')}{non_terminal}'|"


  print("After elimination of left recursion the grammar is: ")
  beta_productions = beta_productions[:-1]
  print(f"{non_terminal} --> {beta_productions}")
  print(f"{non_terminal}'--> {alpha_productions}ε")


non_terminal_symbol = input("Input Non_terminal_symbol: ")
production_split = input("Input Production_split expression: ")
print(f"\nThe Input grammar: {non_terminal_symbol}->{production_split}")

eliminate_left_recursion(non_terminal_symbol, production_split)
```

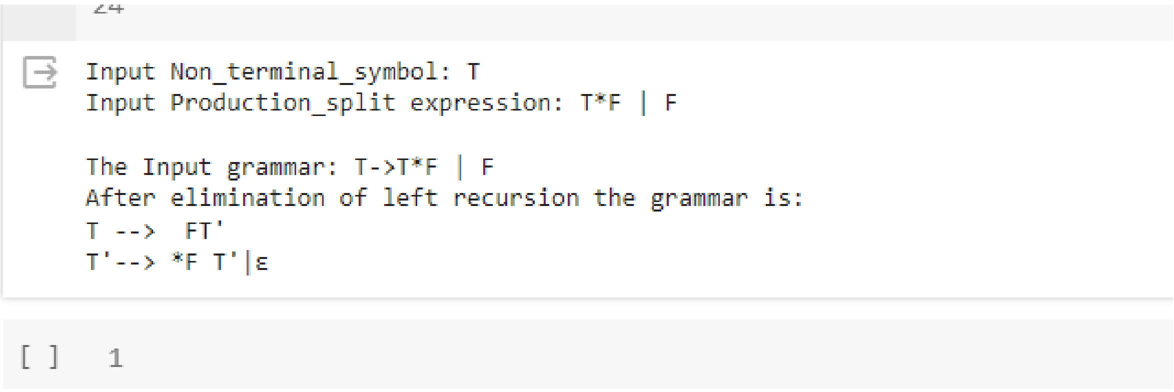
Sample Input:

E->E+T|T
T-> T*F | F



```
Input Non_terminal_symbol: T
Input Production_split expression: T*F | F
```

Observed Output



```
Input Non_terminal_symbol: T
Input Production_split expression: T*F | F

The Input grammar: T->T*F | F
After elimination of left recursion the grammar is:
T -->  FT'
T'--> *F T'|ε
```