University of Asia Pacific

CSE 430

Compiler Design Lab

Lab Exercise 4

Write a Program for Elimination of Left Recursion in a grammar

Name: Md. Azim Islam

Registration: 19201026

Section: A2

Course Code: CSE 430

Semester: Spring 2023

Baivab Das

Lecturer, University of Asia Pacific, Dhaka

CODE

```
from collections import defaultdict
import re

def checkLeftRec(p_rules, sym):
    for rule in p_rules[sym]:
        if re.search("^"+sym, rule):
            return True
    return False



f = open('input', 'r')
p_rules = defaultdict(list)

#input from 'input file'
for line in f:
    if line.strip():
        t1 = line.strip().split("->")
        left = t1[0]
        production = t1[1]
        p_rules[left] += production.split("|")

symbol = list(p_rules.keys())
for i in range(0, len(symbol)):
    new_rules = []
    for j in range(0, i):
        phis = []
        alphas = []
        betas = p_rules[symbol[j]]
        for symbols_ in p_rules[symbol[i]]:
            #if such rule Ai -> Aja + phi
            s = re.search("^"+symbol[j], symbols_)
```

```
            if s:
                #then  Ai -> B1a | B2a | B3a ...+ phi ...
                #given Aj -> B1 | B2 | B3...
                alpha = alphas.append(symbols_[s.end():])
            else:
                phis.append(symbols_)
        if alphas:
            for alpha in alphas:
                for beta in betas:
                    new_rules.append(beta+alpha)
        new_rules += phis
    if new_rules:
        p_rules[symbol[i]] = new_rules

    #Left recursion removal of Ai
    if checkLeftRec(p_rules, symbol[i]):
        alphas = []
        betas = []
        for sym in p_rules[symbol[i]]:
            s = re.search("^"+symbol[i], sym)
            if s:
                alphas.append(sym[s.end():])
            else:
                betas.append(sym)
        new_rules = []
        prime_sym = symbol[i]+"`"
        for beta in betas:
            new_rules.append(beta+prime_sym)
        p_rules[symbol[i]] = new_rules
        new_rules = []
        for alpha in alphas:
            new_rules.append(alpha+prime_sym)
```

```
        p_rules[prime_sym] = new_rules
        if new_rules:
            p_rules[prime_sym].append("ε")


#output
for k in p_rules:
    print(f"{k} -> "+" | ".join(p_rules[k]))
```

INPUT

```
T->T*F|F
```

OUTPUT

```
T -> FT`
T` -> *FT` | ε
```

INPUT

```
E->E+T|T
```

OUTPUT

```
E -> TE`
E` -> +TE` | ε
```