# BCSE307P

# Compiler Design Lab

# Lab Assignment 5


VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI


## Name – Ishan Kapoor

## Registration Number – 21BCE5882


## Submitted to – Prof. S. Srisakthi

Program to eliminate left recursion and left factoring.

## Code:

```
def eliminate_left_recursion(grammar):
    non_terminals = list(grammar.keys())

    for i in range(len(non_terminals)):
        A = non_terminals[i]
        productions = grammar[A]

        for j in range(i):
            B = non_terminals[j]

            new_productions = []
            old_productions = []

            for production in productions:
                if production.startswith(B):
                    new_productions.append(production.replace(B, '', 1))
                else:
                    old_productions.append(production)

            if new_productions:
                new_non_terminal = A + "'"
                grammar[new_non_terminal] = new_productions

                new_productions.append('ε')
                grammar[A] = [production + new_non_terminal for production in
old_productions]
                grammar[new_non_terminal] = [production + new_non_terminal for
production in new_productions]

def left_factor(grammar):
    non_terminals = list(grammar.keys())

    for A in non_terminals:
        productions = grammar[A]

        common_prefixes = {}
        new_productions = []
        old_productions = []

        for production in productions:
            symbol = production[0]
```

```
            if symbol in common_prefixes:
                common_prefixes[symbol].append(production[1:])
            else:
                common_prefixes[symbol] = [production[1:]]

        for symbol, suffixes in common_prefixes.items():
            if len(suffixes) > 1:
                new_non_terminal = A + "'"
                new_productions.append(symbol + new_non_terminal)
                grammar[new_non_terminal] = suffixes
            else:
                old_productions.append(symbol + suffixes[0])

        if new_productions:
            grammar[A] = old_productions
            grammar[A + "'"] = new_productions

# Example usage
grammar = {
    'E': ['E+T', 'T'],
    'T': ['T*F', 'F'],
    'F': ['(E)', 'id']
}

print("Original Grammar:")
print(grammar)

eliminate_left_recursion(grammar)
left_factor(grammar)

print("\nGrammar after eliminating left recursion and left factoring:")
print(grammar)
```

# Output:

```
Original Grammar:
{'E': ['E+T', 'T'], 'T': ['T*F', 'F'], 'F': ['(E)', 'id']}

Grammar after eliminating left recursion and left factoring:
{'E': ['TE\''], 'E\'': ['+TE\'', 'ε'], 'T': ['FT\''], 'T\'': ['*FT\'', 'ε'], 'F': ['(E)', 'id']}
```