**LAB-5**

**2a.Write a program to generate a parse tree for the grammar**

**E → E + T | T**

**T → T \* F | F**

**F → ( E ) | id**

grammar = {

    'E': ['T E\''],

    'E\'': ['+ T E\'', 'ε'],

    'T': ['F T\''],

    'T\'': ['\* F T\'', 'ε'],

    'F': ['( E )', 'id']

}

first = {

    'E': ['(', 'id'],

    'E\'': ['+', 'ε'],

    'T': ['(', 'id'],

    'T\'': ['\*', 'ε'],

    'F': ['(', 'id']

}

follow = {

    'E': [')', '$'],

    'E\'': [')', '$'],

    'T': ['+', ')', '$'],

    'T\'': ['+', ')', '$'],

    'F': ['\*', '+', ')', '$']

}

terminals = ['id', '+', '\*', '(', ')', '$']

non\_terminals = ['E', 'E\'', 'T', 'T\'', 'F']

# Initialize the parse table

parse\_table = {}

# Initialize the parse table with empty values

for nt in non\_terminals:

    parse\_table[nt] = {}

    for t in terminals:

        parse\_table[nt][t] = ''

# Helper function to determine if a symbol is a terminal

def is\_terminal(symbol):

    return symbol in terminals

# Function to fill the parsing table

def fill\_parse\_table():

    for nt in grammar:

        for production in grammar[nt]:

            prod\_first = []

            # If the production starts with a terminal, add it to First set

            first\_symbol = production.split()[0]  # Split production by spaces

            if is\_terminal(first\_symbol):

                prod\_first = [first\_symbol]

            elif first\_symbol == 'ε':

                prod\_first = follow[nt]

            else:

                prod\_first = first[first\_symbol]

            # Fill the parse table based on First set

            for terminal in prod\_first:

                if terminal != 'ε':

                    parse\_table[nt][terminal] = production

            # If ε is in First set, add entries based on Follow set

            if 'ε' in prod\_first:

                for terminal in follow[nt]:

                    parse\_table[nt][terminal] = production

# Fil

fill\_parse\_table()

def print\_parse\_table():

    print(f"{'Non-Terminal':<10} {'|':<2} {' | '.join(terminals)}")

    print("-" \* 60)

    for nt in parse\_table:

        row = f"{nt:<10} | "

        for t in terminals:

            row += f"{parse\_table[nt][t]:<10} | "

        print(row)

print\_parse\_table()

**OUPUT**

