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