## Pandit Deendayal Energy University, Gandhinagar

## School of Technology

**Department of Computer Science & Engineering**

**System Software & Compiler Design Lab (20CP302P)**



# Name: Sutariya Smit Dharmendrabhai

Enrolment No: **21BCP142**

Semester: **V**

Division: **3 (G5)**

Branch: **Computer Science Engineering**

**Practical: 5**

**Aim:** Write a program to calculate first and follow of a given LL (1) grammar.

**Code:**

F = {}

Fo = {}

non\_term = set()

term = set()

# Function to compute FIRST set for a non-terminal

def first\_set(nt):

    if F.get(nt):

        return F[nt]

    F[nt] = set()

    for prod in grammar[nt]:

        for sym in prod:

            if sym in term:

                F[nt].add(sym)

                break

            elif sym == '@':

                F[nt].add('@')

                break

            else:

                F[nt].update(first\_set(sym))

                if '@' not in F[sym]:

                    break

    return F[nt]

# Function to compute FOLLOW set for a non-terminal

def follow\_set(nt):

    if Fo.get(nt):

        return Fo[nt]

    Fo[nt] = set()

    if nt == start\_symbol:

        Fo[nt].add('$')

    for n, prods in grammar.items():

        for prod in prods:

            for i, sym in enumerate(prod):

                if sym == nt:

                    if i < len(prod) - 1:

                        next\_sym = prod[i + 1]

                        if next\_sym in term:

                            Fo[nt].add(next\_sym)

                        else:

                            F\_next = first\_set(next\_sym)

                            Fo[nt].update(F\_next.difference({'@'}))

                            if '@' in F\_next:

                                Fo[nt].update(follow\_set(n))

                    else:

                        Fo[nt].update(follow\_set(n))

    return Fo[nt]

try:

    print("Enter Details of LL1 Grammar.\nEntered Grammar should be LL1")

    t\_count = int(input("Enter the number of terminals: "))

    print("Enter the terminals:")

    term = set(input() for \_ in range(t\_count))

    nt\_count = int(input("Enter the number of non-terminals: "))

    print("Enter the non-terminals:")

    non\_term = set(input() for \_ in range(nt\_count))

    start\_symbol = input("Enter the starting symbol: ")

    p\_count = int(input("Enter the number of productions: "))

    print("Enter the productions in the format NonTerminal -> Production1 | Production2 | ...")

    grammar = {}

    for nt in non\_term:

        grammar[nt] = set()

    for \_ in range(p\_count):

        p\_input = input()

        if '->' in p\_input:

            nt, prods = p\_input.split('->')

            nt = nt.strip()

            prods = prods.split('|')

            grammar[nt] = grammar[nt].union([p.strip() for p in prods])

        else:

            print(f"Invalid production: {p\_input}. Use 'NonTerminal -> Production1 | Production2' format.")

            continue

    # Compute FIRST and FOLLOW sets

    for nt in non\_term:

        first\_set(nt)

        follow\_set(nt)

    print("\nFIRST sets:")

    for nt in non\_term:

        print(f'FIRST({nt}) = {sorted(list(F[nt]))}')

    print("\nFOLLOW sets:")

    for nt in non\_term:

        print(f'FOLLOW({nt}) = {sorted(list(Fo[nt]))}')

except Exception as e:

    print(f"An error occurred: {e}")

**Output:**

