NAME: KSHITIJ GUPTA
Enrolment Number: 21162101007
Sub: CD

Practical – 7[Batch-71]

Code:

```
# Defining the grammar as a dictionary (same as First calculation)
grammar = {
    'S': ['A'],
    'A': ['aBX'],
    'X': ['dX', 'ε'],
    'B': ['b'],
    'C': ['g']
}

# First sets (from previous calculation)
first_sets = {}

# Follow sets dictionary to store results
follow_sets = {}

def find_first(symbol):
    if symbol.islower() and symbol != 'ε':
        return {symbol}

    if symbol in first_sets:
        return first_sets[symbol]
```

```
    first_set = set()

    for production in grammar.get(symbol, []):
        for char in production:
            if char == 'ε':
                first_set.add('ε')
                break
            else:
                char_first = find_first(char)
                first_set.update(char_first - {'ε'})

                if 'ε' not in char_first:
                    break
        else:
            first_set.add('ε')

    first_sets[symbol] = first_set
    return first_set

# Initialize Follow sets for each non-terminal
for non_terminal in grammar:
    follow_sets[non_terminal] = set()

# Add '$' to Follow(S) as S is the start symbol
follow_sets['S'].add('$')

def find_follow(symbol):
```

```
    for lhs in grammar:
        for production in grammar[lhs]:
            for i in range(len(production)):
                if production[i] == symbol:
                    # If there is something after B in A -> αBβ
                    if i + 1 < len(production):
                        next_symbol = production[i + 1]
                        first_of_next = find_first(next_symbol)

                        # Add First(β) to Follow(B) except ε
                        follow_sets[symbol].update(first_of_next - {'ε'})

                        # If First(β) contains ε or B is at the end, add Follow(A) to Follow(B)
                        if 'ε' in first_of_next or i + 1 == len(production) - 1:
                            follow_sets[symbol].update(follow_sets[lhs])

                    # If B is at the end of production, add Follow(A) to Follow(B)
                    if i == len(production) - 1:
                        follow_sets[symbol].update(follow_sets[lhs])

# Compute First sets for all non-terminals
for non_terminal in grammar:
    find_first(non_terminal)

# Compute Follow sets for all non-terminals (iterate multiple times to resolve dependencies)
for _ in range(2):  # Simple two-pass approach (can increase for complex grammars)
    for non_terminal in grammar:
```

find_follow(non_terminal)

Output the Follow sets

print()

for non_terminal, follow_set in follow_sets.items():

print(f"\tfollow({non_terminal}) = {{ {', '.join(follow_set)} }}")

```python
# Defining the grammar as a dictionary (same as First calculation)
grammar = {
    'S': ['A'],
    'A': ['aBX'],
    'X': ['dX', 'ε'],
    'B': ['b'],
    'C': ['g']
}

# First sets (from previous calculation)
first_sets = {}

# Follow sets dictionary to store results
follow_sets = {}

def find_first(symbol):
    if symbol.islower() and symbol != 'ε':
        return {symbol}

    if symbol in first_sets:
        return first_sets[symbol]

    first_set = set()

    for production in grammar.get(symbol, []):
        for char in production:
            if char == 'ε':
                first_set.add('ε')
                break
            else:
```

OUTPUT:

```
PS D:\SEM-7\CD\pr6> & C:/Users/Kshitij/AppData/Roaming/Python/Python311/python.exe d:/SEM-7/CD/pr6/PR7.py

        follow(S) = { $ }
        follow(A) = { $ }
        follow(X) = { $ }
        follow(B) = { d, $ }
        follow(C) = {  }
PS D:\SEM-7\CD\pr6>
```