Ganpat University
॥ विद्यया समाजोत्कर्षः ॥
Institute of Computer Technology

Name: Tushar Panchal

En.No: 21162101014

Sub: CD(Compiler Design)

Branch: CBA

Batch:71

----------PRACTICAL 07----------

Write a Program to find FOLLOW of Given Grammar.

S -> A
A -> aBX
X -> dX | ε
B -> b
c -> g

✓ 7.py :

```
# Grammar definition
productions = {
    'S': ['A'],
    'A': ['aBX'],
    'X': ['dX', 'ε'],  # 'ε' represents the empty string
    'B': ['b'],
    'c': ['g']
}

# First sets from the previous program (assumed to be calculated)
first_sets = {
    'S': {'a'},
    'A': {'a'},
    'X': {'d', 'ε'},
    'B': {'b'},
    'c': {'g'}
}

# Dictionary to store follow sets
```

```
follow_sets = {non_terminal: set() for non_terminal in productions}

# Start symbol should have '$' in its follow set
follow_sets['S'].add('$')

# Function to calculate First of a string (could be terminals or non-
terminals)
def find_first_of_string(symbols):
    first = set()
    for symbol in symbols:
        if symbol.islower():  # Terminal symbol
            first.add(symbol)
            break
        elif symbol == 'ε':  # Epsilon
            first.add('ε')
            break
        else:
            first_of_symbol = first_sets[symbol]
            first.update(first_of_symbol - {'ε'})
            # If First of symbol contains ε, continue to the next
symbol
            if 'ε' not in first_of_symbol:
                break
    else:
        # If all symbols can derive ε, add ε to the First of string
        first.add('ε')
    return first

# Function to calculate the Follow set of each non-terminal
def compute_follow_sets():
    changed = True
    while changed:
        changed = False
        for non_terminal, rules in productions.items():
            for rule in rules:
                trailer = follow_sets[non_terminal].copy()
                for i in range(len(rule) - 1, -1, -1):
                    symbol = rule[i]
                    if symbol.isupper():  # If symbol is a non-terminal
                        if follow_sets[symbol].update(trailer):
                            changed = True
                        # If First of the next part contains ε, add
follow of current non-terminal to trailer
                        if 'ε' in first_sets[symbol]:
                            trailer.update(first_sets[symbol] - {'ε'})
                        else:
                            trailer = first_sets[symbol]
                    elif symbol.islower():
```

```
                        trailer = {symbol}

# Main function to run the program
if __name__ == '__main__':
    compute_follow_sets()
    # Output the Follow sets
    for non_terminal, follow in follow_sets.items():
        print(f"Follow({non_terminal}) = {{ {', '.join(follow)} }}")
```

✓ **Output :**