

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CD Practical 7

Aim : Write a Program to find Follow of Given Grammar.

S \rightarrow A

A \rightarrow aBX

X \rightarrow dX | ϵ

B \rightarrow b

c \rightarrow g

Code :

```
# Defining the grammar as a dictionary (same as First calculation)
grammar = {
    'S': ['A'],
    'A': ['aBX'],
    'X': ['dX', ' $\epsilon$ '],
    '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  $\neq$  ' $\epsilon$ ':
        return {symbol}

    if symbol in first_sets:
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CD Practical 7

```
        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):
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CD Practical 7

```
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  $A \rightarrow \alpha B \beta$ 
                if i + 1 < len(production):
                    next_symbol = production[i + 1]
                    first_of_next = find_first(next_symbol)

                    # Add First( $\beta$ ) to Follow(B) except  $\epsilon$ 
                    follow_sets[symbol].update(first_of_next - {' $\epsilon$ '})

                    # If First( $\beta$ ) contains  $\epsilon$  or B is at the end, add
                    Follow(A) to Follow(B)
                    if ' $\epsilon$ ' 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)
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CD Practical 7

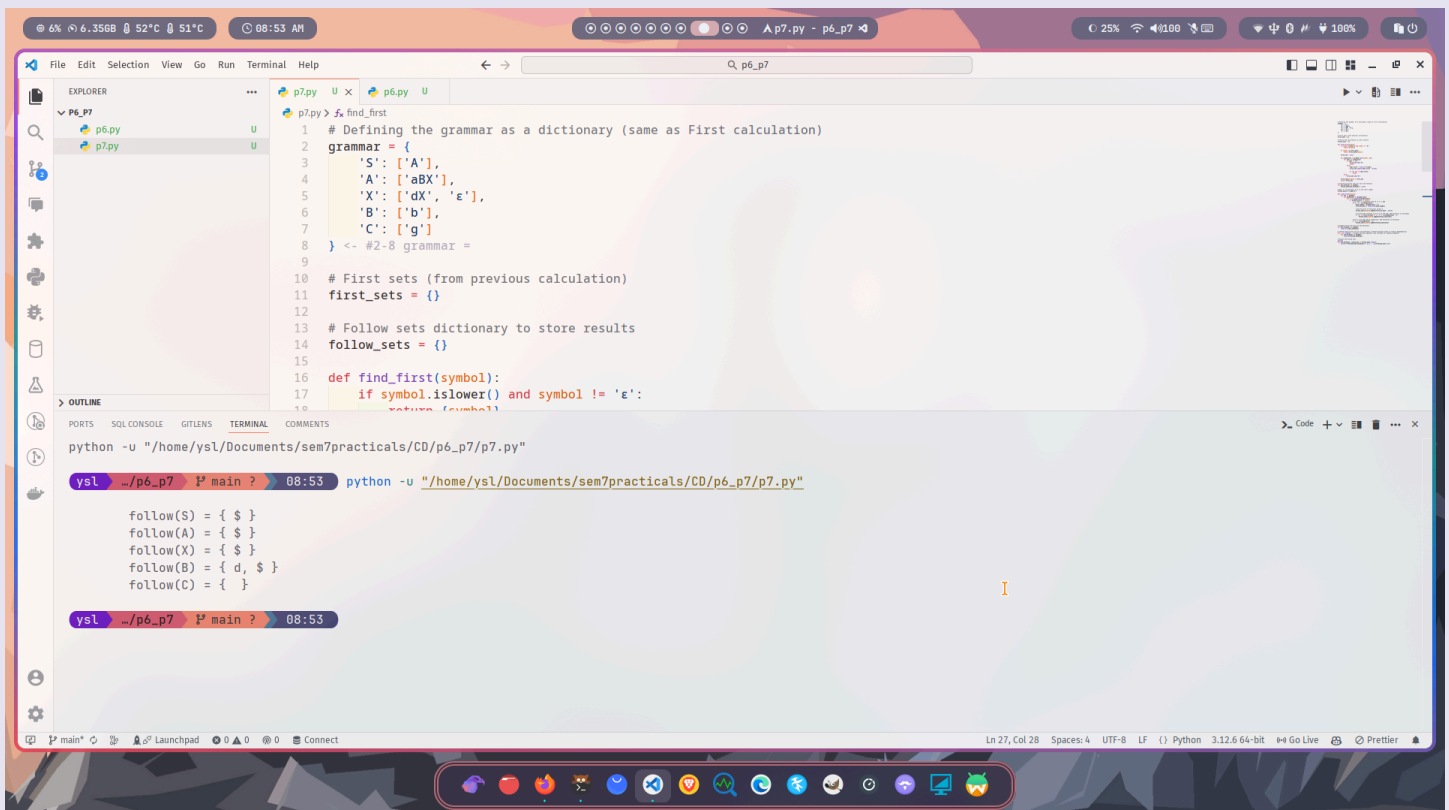
```
for _ in range(2): # Simple two-pass approach (can increase for complex grammars)

    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)} }}" )
```

Output :



The screenshot shows a VS Code editor window with a Python script named `p7.py` and its output in the terminal. The script defines a grammar and calculates the Follow sets for non-terminals A, B, and C. The terminal output shows the results of the `find_follow` function.

```
p7.py 1 # Defining the grammar as a dictionary (same as First calculation)
2 grammar = {
3     'S': ['A'],
4     'A': ['aBX'],
5     'X': ['dX', 'ε'],
6     'B': ['b'],
7     'C': ['g']
8 } <- #2-8 grammar =
9
10 # First sets (from previous calculation)
11 first_sets = {}
12
13 # Follow sets dictionary to store results
14 follow_sets = {}
15
16 def find_first(symbol):
17     if symbol.islower() and symbol != 'ε':
18         return [symbol]
```

```
python -u "/home/ysl/Documents/sem7practicals/CD/p6_p7/p7.py"

follow(S) = { $ }
follow(A) = { $ }
follow(X) = { $ }
follow(B) = { d, $ }
follow(C) = { }
```