

**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  $A \rightarrow \alpha B \beta$**

**if  $i + 1 < \text{len}(\text{production})$ :**

**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  $i + 1 == \text{len}(\text{production}) - 1$ :**

**follow\_sets[symbol].update(follow\_sets[lhs])**

**# If B is at the end of production, add Follow(A) to Follow(B)**

**if  $i == \text{len}(\text{production}) - 1$ :**

**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:**

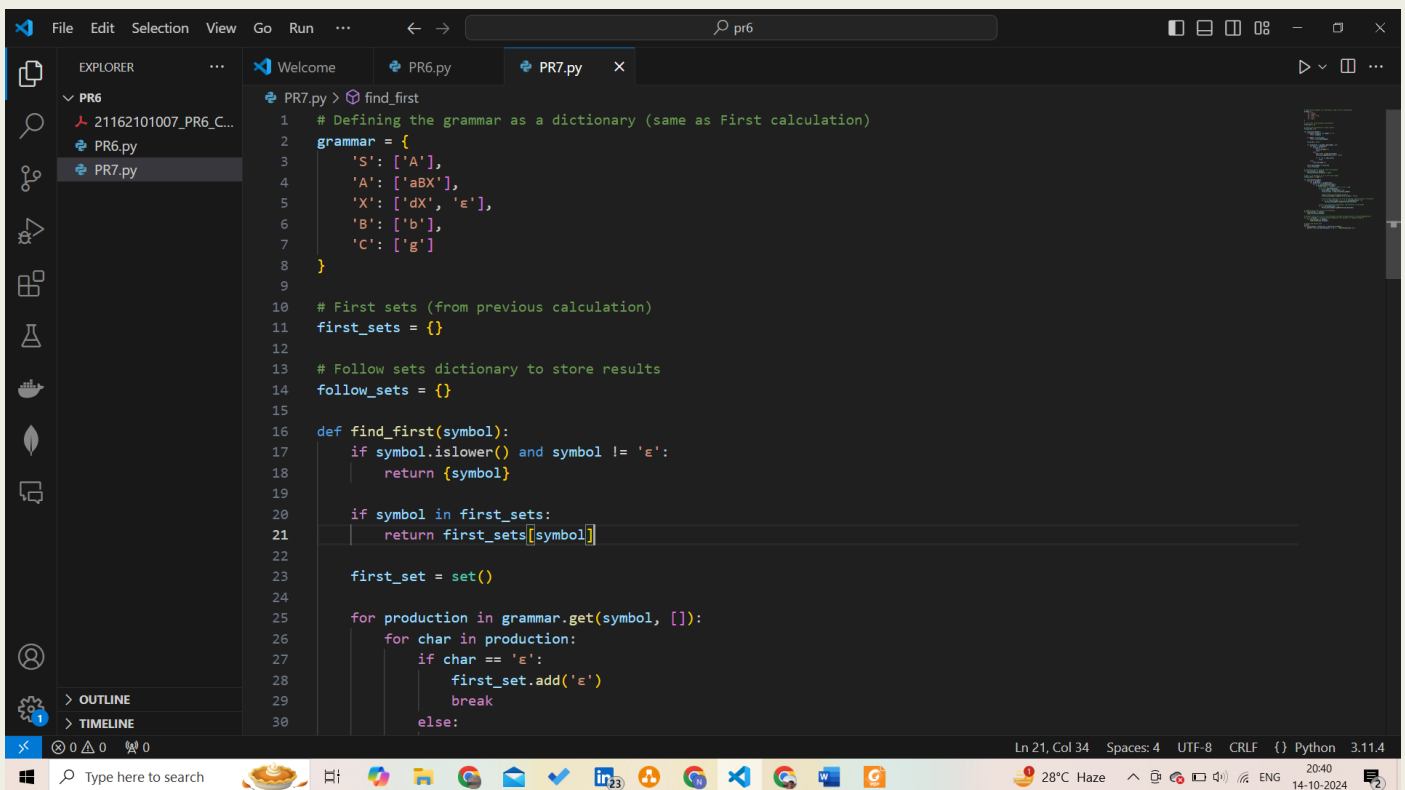
```
find_follow(non_terminal)
```

# Output the Follow sets

```
print()
```

```
for non_terminal, follow_set in follow_sets.items():
```

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



The screenshot shows a VS Code editor with a Python file named PR7.py. The code defines a grammar and calculates follow sets. The grammar is defined as a dictionary with non-terminals 'S', 'A', 'X', 'B', and 'C' and their respective productions. The follow sets are calculated using a function find\_first, which returns the first set for a given symbol. The follow sets are then calculated for each non-terminal in the grammar.

```
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 }
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}
19
20     if symbol in first_sets:
21         return first_sets[symbol]
22
23     first_set = set()
24
25     for production in grammar.get(symbol, []):
26         for char in production:
27             if char == 'ε':
28                 first_set.add('ε')
29                 break
30             else:
```

OUTPUT:



The screenshot shows a terminal window with the output of the follow set calculation. The output shows the follow sets for each non-terminal in the grammar.

```
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>
```