□ Low-Level Design (LLD) for Temperature Converter

Difficulty Level: Easy | Total Marks: 10

Standards Followed: 4 Functions | 3 Visible Test Cases | 2 Hidden Test Cases

□ Concepts Tested

□ NumPy array creation and transformation
□ Mathematical operations and rounding
□ Categorization with conditionals
□ OOP with method-based class design

□ Problem Statement

Design a system that manages a list of Celsius temperatures and performs operations such as:
• Loading temperatures
• Converting them to Fahrenheit
• Calculating the average
• Categorizing each temperature as "Cold", "Moderate", or "Hot"

□ Operations

1. Load Temperatures
Function Prototype:
def load_temperatures(self, temp_list: list):
Example Input:
load_temperatures([10, 20, 30])
Expected Output:
[10. 20. 30.]

2. Convert to Fahrenheit
Function Prototype:
def convert_to_fahrenheit(self):
Expected Output:
[50. 68. 86.]

3. Calculate Average
Function Prototype:
def calculate_average(self):
Expected Output:
20.0

4. Categorize Temperatures
Function Prototype:
def categorize_temperatures(self):
Expected Output:
['Cold', 'Moderate', 'Hot']

□ Implementation Code

```
import numpy as np

class TemperatureConverter:
    def __init__(self):
        """Initializes an empty temperature array."""
        self.temps = np.array([])

    def load_temperatures(self, temp_list):
        """Loads a list of Celsius temperatures."""
        self.temps = np.array(temp_list, dtype=float)
        print(self.temps)

    def convert_to_fahrenheit(self):
        """Converts temperatures to Fahrenheit."""
        fahrenheit = np.round((self.temps * 9 / 5) + 32, 1)
        print(fahrenheit)

    def calculate_average(self):
        """Calculates and prints average Celsius temperature."""
        print(round(np.mean(self.temps), 1))

    def categorize_temperatures(self):
        """Categorizes temperatures: Cold <10, Moderate 10–25, Hot >25."""
        categories = []
        for t in self.temps:
            if t < 10:
                categories.append("Cold")
            elif t <= 25:
                categories.append("Moderate")
            else:
                categories.append("Hot")
        print(categories)

# Driver Code
if __name__ == "__main__":
    tc = TemperatureConverter()
```

```
q = int(input())
for _ in range(q):
    cmd = input().split()
    if cmd[0] == "load":
        tc.load_temperatures(list(map(float, cmd[1:])))
    elif cmd[0] == "convert":
        tc.convert_to_fahrenheit()
    elif cmd[0] == "average":
        tc.calculate_average()
    elif cmd[0] == "categorize":
        tc.categorize_temperatures()
    else:
        print("Invalid operation")
```

Test Case ID	Test Case Description	Associated Function(s)	Marks
TC1	Loading temperatures and printing the array	load_temperatures()	□ 2 Marks
TC2	Converting Celsius to Fahrenheit	convert_to_fahrenheit()	□ 2 Marks
TC3	Calculating average temperature	calculate_average()	□ 2 Marks
HTC1	Categorizing edge values (10, 25, 26)	categorize_temperatures()	□ 2 Marks
HTC2	Handling cold temperature only (e.g. below 10°C)	load_temperatures(), categorize_temperatures()	□ 2 Marks
TOTAL	All test cases passed	-	□ **10 Marks**

□ Visible Test Cases

TC1 Input:
1
load 10 20 30
Output:
[10. 20. 30.]

TC2 Input:
1
convert
Output:
[50. 68. 86.]

TC3 Input:
1
average
Output:
20.0

□ Hidden Test Cases

HTC1 Input:
1
categorize
Output:
['Moderate', 'Moderate', 'Hot']

HTC2 Input:
2
load 5
categorize
Output:
[5.]
['Cold']