Lab 3 report

In-Lab Tasks:

Task 1:

```
# Function definition
def count_even_numbers(number_list):
    # Initialize counter variable
    even_count = 0
    # Loop through each number in the list
    for num in number_list:
        # Check if the number is even
        if num % 2 == 0:
            even_count += 1
    # Return the count of even numbers
    return even_count
# Test cases
numbers1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
numbers2 = [11, 13, 15, 17]
numbers3 = []  # Empty list
# Display results
print(count_even_numbers(numbers1))  # Expected output: 5
print(count_even_numbers(numbers2))  # Expected output: 0
print(count_even_numbers(numbers3))  # Expected output: 0
```

Output:

```
In [1]: runfile('C:/Users/malik/OneDrive/Documents/GitHub/FA20-BCE-024_AL_LAB/lab3/
inLabTask1.py', wdir='C:/Users/malik/OneDrive/Documents/GitHub/FA20-BCE-024_AL_LAB/lab3')
5
0
0
```

Task 2:

```
def calculate_gpa(students):
    # Dictionary to map percentage ranges to grade points
    grade_point_mapping = {
        (85, 100): 4.00,
        (80, 84): 3.66,
        (75, 79): 3.33,
        (71, 74): 3.00,
        (68, 70): 2.66,
        (64, 67): 2.33,
        (61, 63): 2.00,
        (58, 60): 1.66,
        (54, 57): 1.30,
        (50, 53): 1.00,
        (0, 49): 0.00,
    }
    result = []

    for student in students:
        name = student['name']
        marks = student['marks']

        # Calculate the grade point for each course
        grade_points = []
        for mark in marks:
            percentage = (mark / 100) * 100

            # Determine the grade point based on the percentage
            grade_point = None
            for (min_range, max_range), gp in grade_point_mapping.items():
                if min_range <= percentage <= max_range:
                    grade_point = gp
                    break

            grade_points.append(grade_point)

        # Calculate GPA by averaging the grade points
        gpa = sum(grade_points) / len(grade_points)

        # Create the student's record
        student_record = {
            'name': name,
            'grades': marks,
            'grade_points': grade_points,
            'gpa': gpa
        }
```

```
        result.append(student_record)
    return result
# Example student records
students = [
    {'name': 'Ali', 'marks': [75, 88, 75, 83, 82]},
    {'name': 'Babar', 'marks': [67, 77, 75, 69, 65]},
    {'name': 'Uzair', 'marks': [81, 68, 78, 82, 84]},
    {'name': 'Eman', 'marks': [91, 86, 69, 72, 61]},
    {'name': 'Sidra', 'marks': [88, 56, 71, 50, 51]}
]
# Calculate GPA for the students
gpa_results = calculate_gpa(students)
# Print the results
for student in gpa_results:
    print(f"Name: {student['name']}")
    print(f"Grades: {student['grades']}")
    print(f"Grade Points: {student['grade_points']}")
    print(f"GPA: {student['gpa']}")
    print()
```

Output:

```
In [12]: runfile('C:/Users/malik/OneDrive/Documents/GitHub/FA20-BCE-024_AL_LAB/lab3/
inLabTask2.py', wdir='C:/Users/malik/OneDrive/Documents/GitHub/FA20-BCE-024_AL_LAB/lab3')
Name: Ali
Grades: [75, 88, 75, 83, 82]
Grade Points: [3.33, 4.0, 3.33, 3.66, 3.66]
GPA: 3.596

Name: Babar
Grades: [67, 77, 75, 69, 65]
Grade Points: [2.33, 3.33, 3.33, 2.66, 2.33]
GPA: 2.7960000000000003

Name: Uzair
Grades: [81, 68, 78, 82, 84]
Grade Points: [3.66, 2.66, 3.33, 3.66, 3.66]
GPA: 3.3939999999999997

Name: Eman
Grades: [91, 86, 69, 72, 61]
Grade Points: [4.0, 4.0, 2.66, 3.0, 2.0]
GPA: 3.132

Name: Sidra
Grades: [88, 56, 71, 50, 51]
Grade Points: [4.0, 1.3, 3.0, 1.0, 1.0]
GPA: 2.06
```

Task 3:

```
class Student:
    def __init__(self, name, roll_number):
        self.name = name
        self.roll_number = roll_number
        self.marks = []

    def add_marks(self, subject, score):
        self.marks.append((subject, score))

    def calculate_average(self):
        if not self.marks:
            return 0  # Return 0 if there are no marks
        total_score = sum(score for _, score in self.marks)
        return total_score / len(self.marks)

# Create an instance of the Student class
student1 = Student("M.Husnian", "101")

# Add marks for different subjects
student1.add_marks("Science", 79)
student1.add_marks("Math", 89)
student1.add_marks("History", 78)
student1.add_marks("English", 88)
student1.add_marks("Art", 90)

# Calculate and print the average marks
average_marks = student1.calculate_average()
print(f"Average Marks for {student1.name} with (Roll Number: {student1.roll_number}):
{average_marks:.2f}")
```

Output:

```
In [20]: runfile('C:/Users/malik/OneDrive/Documents/GitHub/FA20-BCE-024_AL_LAB/lab3/
inLabTask3.py', wdir='C:/Users/malik/OneDrive/Documents/GitHub/FA20-BCE-024_AL_LAB/lab3')
Average Marks for M.Husnian with (Roll Number: 101): 84.80
```

Post Lab:

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author
        self.available = True

    def borrow(self):
```

```
        if self.available:
            self.available = False
            return f"You have successfully borrowed '{self.title}' by {self.author}."
        else:
            return f"'{self.title}' by {self.author} is already borrowed."

    def return_book(self):
        if not self.available:
            self.available = True
            return f"You have returned '{self.title}' by {self.author}."
        else:
            return f"'{self.title}' by {self.author} is already available."

# Create some book instances
book1 = Book("Peer-e-Kamil", "Umera Ahmed")
book2 = Book("Maps for Lost Lovers", "Nadeem Aslam")
book3 = Book("Home Fire", " Kamila Shamsie")

# Borrow and return books
print(book1.borrow())  # Borrow 'Peer-e-Kamil'
print(book1.borrow())  # Already borrowed
print(book1.return_book())  # Return 'Peer-e-Kamil'
print(book2.borrow())  # Borrow 'To Kill a Mockingbird'
print(book3.borrow())  # Borrow 'Home Fire'
print(book1.return_book())  # Already available
```

Output:

```
In [22]: runfile('C:/Users/malik/OneDrive/Documents/GitHub/FA20-BCE-024_AL_LAB/lab3/
postLab.py', wdir='C:/Users/malik/OneDrive/Documents/GitHub/FA20-BCE-024_AL_LAB/lab3')
You have successfully borrowed 'Peer-e-Kamil' by Umera Ahmed.
'Peer-e-Kamil' by Umera Ahmed is already borrowed.
You have returned 'Peer-e-Kamil' by Umera Ahmed.
You have successfully borrowed 'Maps for Lost Lovers' by Nadeem Aslam.
You have successfully borrowed 'Home Fire' by  Kamila Shamsie.
'Peer-e-Kamil' by Umera Ahmed is already available.
```