**Python Assessment - 2**

**1. Object-Oriented Programming (OOP)**

**Problem Statement:**

Create a class `Circle` that represents a circle with a radius. Implement methods to calculate the area and circumference of the circle.

**Example:**

circle = Circle(5)

print(circle.area()) # Output: 78.53975

print(circle.circumference()) # Output: 31.4159

**2. File Handling**

**Problem Statement:**

Write a function `read\_file\_lines(filename: str)` that reads a file and returns its contents as a list of lines.

**Example:**

lines = read\_file\_lines('example.txt')

print(lines) # Output: ['First line', 'Second line', ...]

**3. Error Handling & Exception**

**Problem Statement:**

Create a function `safe\_divide(a: float, b: float)` that divides two numbers. Handle division by zero gracefully by returning `None`.

**Example:**

print(safe\_divide(10, 2)) # Output: 5.0

print(safe\_divide(10, 0)) # Output: None

**4. Database Operations with MySQL**

**Problem Statement:**

Write a program that connects to a MySQL database. Implement the following functionalities:

1. Create a table named `users` with fields `id` (auto-increment) and `name` (string).

2. Insert a user into the `users` table.

3. Fetch and print all users from the `users` table.

**Example:**

# Create the database and table

create\_database\_and\_table()

# Insert a user

insert\_user('Alice')

# Fetch and print all users

users = fetch\_all\_users()

print(users) # Output: [(1, 'Alice')]

Note: Make sure to handle database connection and exceptions appropriately.

**5. Functions & Modules**

**Problem Statement:**

Create a module-level function `factorial(n: int)` that computes the factorial of a number using recursion.

**Example:**

print(factorial(5)) # Output: 120

**6. Data Structures**

**Problem Statement:**

Implement a function `count\_elements(lst: List[int])` that counts the occurrences of each element in a list and returns a dictionary.

**Example:**

result = count\_elements([1, 2, 2, 3, 3, 3])

print(result) # Output: {1: 1, 2: 2, 3: 3}

**7. Control Flow - Conditional Statement**

**Problem Statement:**

Write a function `grade(score: float)` that returns the letter grade for a given score based on the following scale:

- A: 90-100

- B: 80-89

- C: 70-79

- D: 60-69

- F: 0-59

Example:

print(grade(85)) # Output: B

**8. Control Flow - Looping**

**Problem Statement:**

Create a function `fibonacci(n: int)` that returns the first `n` numbers in the Fibonacci sequence.

**Example:**

print(fibonacci(5)) # Output: [0, 1, 1, 2, 3]

**9. File Handling - Writing**

**Problem Statement:**

Implement a function `write\_lines\_to\_file(filename: str, lines: List[str])` that writes a list of strings to a specified file, each on a new line.

**Example:**

write\_lines\_to\_file('output.txt', ['Line 1', 'Line 2'])

# This will create 'output.txt' containing:

# Line 1

# Line 2

**10. Filtering Armstrong Numbers**

**Problem Statement:**

Write a function `filter\_armstrong\_numbers(numbers: List[int])` that filters and returns a list of Armstrong numbers from the given list.

**Example:**

result = filter\_armstrong\_numbers([153, 123, 370, 1, 407])

print(result) # Output: [153, 370, 407]

**11. Sorting Algorithms**

**Problem Statement:**

Implement two functions: `bubble\_sort(arr: List[int])` and `insertion\_sort(arr: List[int])` that sort a list of integers using bubble sort and insertion sort algorithms, respectively.

**Example:**

arr = [64, 34, 25, 12, 22]

bubble\_sort(arr)

print(arr) # Output: [12, 22, 25, 34, 64]

insertion\_sort(arr)

print(arr) # Output: [12, 22, 25, 34, 64]

**12. Reversing a String**

**Problem Statement:**

Write a function `reverse\_string(s: str)` that reverses a given string without using built-in functions or slicing.

**Example:**

result = reverse\_string("hello")

print(result) # Output: "olleh"

**13. Unique Elements in a List**

**Problem Statement:**

Create a function `unique\_elements(lst: List[int])` that returns a list of unique elements from the given list.

**Example:**

result = unique\_elements([1, 2, 2, 3, 4, 4, 5])

print(result) # Output: [1, 2, 3, 4, 5]

**14. Floyd's Triangle**

**Problem Statement:**

Write a function `floyds\_triangle(n: int)` that prints Floyd's Triangle up to `n` rows.

**Example:**

floyds\_triangle(5)

# Output:

# 1

# 2 3

# 4 5 6

# 7 8 9 10

# 11 12 13 14 15

**15. Diamond Pattern**

**Problem Statement:**

Create a function `diamond\_pattern(n: int)` that prints a diamond pattern of stars for `n` rows.

**Example:**

diamond\_pattern(5)

# Output:

# \*

# \*\*\*

# \*\*\*\*\*

# \*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*

# \*\*\*\*\*

# \*\*\*

# \*

**Instructions:**

1. Implement each function as described in the problem statement.

2. Make sure to test your functions with various inputs.

3. Use comments to explain your code where necessary.