National University of Computer and Emerging Sciences

OOP (CS1004)

Date: May 10th 2024

Course Instructor(s)

Ms. Mahnoor, Ms. Fatima, Ms. Shaharbano

Lab Final Exam (B)

Total Time: 2 Hours

Total Marks: 100

Total Questions: 03

Semester: SP-2024

Campus: Karachi

Dept. Computer Science

Student Name | Roll No | Section | Student Signature

Question Q1 ***Weightage: 15;*** **Marks: 30; CLO 2**

You've been tasked with developing a comprehensive library management system that caters to various types of books, including novels and textbooks.

Your task involves designing an abstract class named `Book` to serve as the foundation for managing different book items efficiently. The `Book` class includes essential attributes such as `book_id`, `title`, `author`, `price`, and `quantity`, with the `title` attribute being an array of pointers to book titles. Moreover, it outlines the blueprint for essential methods like `add_book`, `remove_book`, and `update_book`.

Expanding upon the `Book` class, there are two specialized classes: `Novel` and `Textbook`. The `Novel` class extends `Book` and introduces unique features such as `genre`, `publication_year`, and `rating`. Similarly, the `Textbook` class builds upon `Book` and incorporates attributes like `subject`, `edition`, and `publisher`. Both the `Novel` and `Textbook` classes meticulously override the methods inherited from the `Book` class, tailoring them to the specific requirements of novel and textbook management.

Your challenge is to utilize polymorphism and the array of pointers within the `Book` class to implement these methods efficiently across different categories of books.

Question Q2 ***Weightage: 15;*** **Marks: 30; CLO 3**

Consider a scenario where you're tasked with designing a class named `SecureDataVault` to manage sensitive information with utmost security. This class comprises private attributes: `data_storage`, an array storing encrypted sensitive data i-e simple message ("Let's meet on Friday") that is stored dynamically, `data_sizes`, storing the size of string, and `encryption_key`, a key crucial for encryption purposes. To interact with the data vault securely, three public methods, `store_data`, `retrieve_data`, and `erase_data`, are provided. The `store_data` method encrypts the data using the encryption key before storing it in the array.

Additionally, you are required to create a specialized class named `SecurityAgent` with exclusive access to the private attributes and methods of the `SecureDataVault` class. This `SecurityAgent` class is entrusted with the responsibility of safeguarding the integrity and confidentiality of the data stored in the vault and providing necessary access to private data of the `SecureDataVault` class.

The `SecurityAgent` class has private attributes `agent_name` and `decryption_key`, serving as identifiers for the agent and the decryption key, respectively.

The `SecurityAgent` class is assigned the following responsibilities:

- Access and print the attributes of the `SecureDataVault`.

- If provided with an empty string, update the attributes of the vault using the `store_data` method using (string + encrypted_key) % 26 formula.

- Execute complex operations on the data stored in the vault, such as decryption or analysis, by implementing them as methods. (string - encrypted_key) % 26

Your challenge is to design the `SecurityAgent` class to provide controlled access to the private attributes and methods of the `SecureDataVault`, ensuring the maintenance of data integrity and confidentiality at all times. along with proper error handling mechanisms. Ensure that dynamic memory allocation is used for the arrays `data_storage` and `data_sizes` in the `SecureDataVault` class to accommodate varying amounts of data. Implement error handling to address potential issues such as out-of-bounds access, invalid input, and memory allocation failures.

Question Q3 ***Weightage: 20;*** **Marks: 40; CLO: 4**

You have to craft a generic class hierarchy to manage information about individuals engaged in academic pursuits. The foundational class is termed `Person`, which encapsulates attributes like `Name`, `Age`, and `Role`. It furnishes the subsequent functionalities:

- A method designated `getName`, returning the individual's name.

- A method labeled `getAge`, providing the individual's age.

- A method dubbed `getRole`, furnishing the individual's role.

- A method named `getSalary`, offering the individual's salary.

A default constructor for `Person` is available, initializing all variables to default values, handled with templates. In addition, there exist two generic derived classes:

1. `TA`: Inherits from the superclass `Person`, with an overridden function `getRole () ` to supply specific details pertaining to the student's academic engagement. Attributes of this class encompass `Program`, `GPA`, `CoursesIncharge`, `CreditsEarned`, etc.

2. `Professor`: Also inheriting from the `Person` class, `Professor` further overrides the `getRole () ` function to furnish details about the academic position held by the professor. Parameters within this class include `Department`, `Position`, `TeachingExperience`, `ResearchProjects`, etc.

Both `TA` and `Professor` classes possess parameterized constructors and accessor/mutator functions for their attributes. Additionally, a display function is provided to exhibit the details of each class.

Moreover, special functions are implemented for both `TA` and `Professor` classes to compute and return the salary. These functions utilize operator overloading for pre and post increment operations, augmenting the salary by 10% and 5%, respectively, based on any distinctive characteristic.