CS3201: OBJECT ORIENTED PROGRAMMING LABORATORY

Topic: Multiple Inheritance, Function Overriding, Constructors in Inheritance, Memory Management

Lab: 5

Date: March 12, 2025

Spot Questions:

1. Create a base class **ComplexNumber** with attributes *real* and *imag*, and *overload operator** to perform complex number multiplication. Create another base class **MathUtilities** with a method *magnitude()* to compute the magnitude of a complex number and a method *conjugate()* to return the complex conjugate. Derive a class **AdvancedComplex** that inherits from both, overrides operator* to improve multiplication accuracy using the conjugate method, and uses a dynamically allocated char* to store multiplication history. Initialize all attributes via constructor chaining, manage memory properly, and test with multiple complex number multiplications and magnitude calculations in *main()*.

2. Create a base class **CartItem** with attributes *itemName* and *price*, and a virtual method *calculateFinalPrice()* to compute the final price after applying discounts or taxes. Create derived classes **ElectronicsItem**, **ClothingItem**, and **GroceryItem**, each overriding *calculateFinalPrice()* with specific pricing rules: *ElectronicsItem* applies a **5% discount** if the price is over Rs.5000/-, *ClothingItem* applies a **seasonal discount**, and *GroceryItem* adds **dynamic tax**. Use a base class pointer to store different item types in a vector<**CartItem***> and process them dynamically. Test in *main()* by adding multiple items to the cart and verifying that the correct overridden method is executed for each type.

3. Create a base class **Student** with attributes *name* and *ID*, and a constructor that initializes them. Derive a class **GradedStudent** from Student, adding an attribute score. Implement constructor chaining so that **GradedStudent** calls the **Student** constructor before initializing score. Overload the "< *and* ==" operators to compare students based on their scores. Test in *main()* by creating <u>multiple students and comparing them to determine rankings</u>.

4. Create a base class **Passenger** with attributes char* *name*, int *age*, and char* *passportNumber*, with memory allocated dynamically for *name* and *passportNumber*. Implement a copy constructor and overloaded assignment operator to ensure **deep copying**. Derive a class **Ticket** that inherits from Passenger, adding char* *flightNumber* and double price, with *flightNumber* dynamically allocated. Ensure the destructor correctly deallocates all dynamically allocated memory. Test in *main()* by creating and copying Ticket objects to verify correct memory management and deep copy handling.