Object Oriented Programming

Practice Task

Task # 1

Create a class **HugeInteger** that uses a dynamically created 40- element array of characters to store integers as large as 40 digits each. Use operator overloading to provide functionality for **input**, **output**, **add** and **subtract**. For comparing HugeInteger objects, provide overloaded operators which perform the functionality: **isEqualTo, isNotEqualTo, isGreaterThan, isLessThan, isGreaterThanOrEqualTo, and isLessThanOrEqualTo —** each of these is a "predicate" functionality (a function that simply returns true if the relationship holds between the two HugeIntegers and returns false if the relationship does not hold). Also, provide a predicate function isZero. Use operator overloading to provide functionality multiply, modulus and divide too.

Task # 2

Library System – Make a Library management system. Books will be added through the interface of your program (books can be deleted as well). You will be able to save the books with respect to author's name, book title and publisher name. Each book will be allotted a unique book code. The record can be searched. All books can be displayed (through a table) with respect to author's name, book title and publisher's name. Books can be issued to people and can be returned by the people. The record must be updated after issuance or returning. There can be more than 1 copy of a particular book with each copy having a unique code. Library membership records must also be maintained. New members can be added. Members can be deleted as well. All members must have unique ID. Members must be categorized into 3 categories: Gold, Silver and Bronze. Gold members can get 5 books issued at a time. Silver members can get at most 3 books and Bronze members get only 1 book issued from the library. List of library members can also be displayed in alphabetical order or with respect to their ID or membership status. Members can also be searched. All books must be issued for a period of 2 weeks. Your program should save the date of issuance. There must be a function to change the current date. There must be a function which displays information of all those members who have to return the books if the current date is equal to the date of returning on that particular day.
[You have to design your system according to above requirements. Also identify the classes and attribute from given requirements.]

Task # 3

(Package *Inheritance Hierarchy*) Package-delivery services, such as FedEx®, DHL® and UPS®, offer a number of different shipping options, each with specific costs associated. Create an inheritance hierarchy to represent various types of packages. Use class Package as the base class of the hierarchy, then include classes TwoDayPackage and OvernightPackage that derive from Package. Base class Package should include data members representing the name, address, city, state and ZIP code for both the sender and the recipient of the package, in addition to data members that store the weight (in ounces) and cost per ounce to ship the package. Package's constructor should initialize these data members. Ensure that the weight and cost per ounce contain positive values. Package should provide a public member function calculateCost that returns a double indicating the cost associated with shipping the package. Package's calculateCost function should determine the cost by multiplying the weight by the cost per ounce. Derived class TwoDayPackage should inherit the functionality of base class Package, but also include a data member that represents a flat fee that the shipping company charges for two-day-delivery service. TwoDayPackage's constructor should receive a value to initialize this data member. TwoDayPackage should redefine member function calculate-
Cost so that it computes the shipping cost by adding the flat fee to the weight-based cost calculated by base class Package's calculateCost function. Class OvernightPackage should inherit directly from class Package and contain an additional data member representing an additional fee per ounce charged for overnight-delivery service. OvernightPackage should redefine member function calcu lateCost so that it adds the additional fee per ounce to the standard cost per ounce before calculating the shipping cost. Write a test program that creates objects of each type of Package and test member function calculateCost.