1. (`Rectangle` Class) Create a class `Rectangle` with attributes `length` and `width`, each of which defaults to 1. Provide member functions that calculate the `perimeter` and the `area` of the rectangle. Also, provide set and get functions for the `length` and `width` attributes. The set functions should verify that `length` and `width` are each floating-point numbers larger than 0.0 and less than 20.0.

2. (Invoice Class) Create a class called Invoice that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as data members a part number (type string), a part description (type string), a quantity of the item being purchased (type int) and a price per item (type int). Your class should have a constructor that initializes the four data members. Provide a set and a get function for each data member. In addition, provide a member function named getInvoiceAmount that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as an int value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0. Write a test program that demonstrates class Invoice's capabilities.

3. (Account Class) Create a class called Account that a bank might use to represent customers' bank accounts. Your class should include one data member of type int to represent the account balance. Your class should provide a constructor that receives an initial balance and uses it to initialize the data member. The constructor should validate the initial balance to ensure that it is greater than or equal to 0. If not, the balance should be set to 0 and the constructor should display an error message, indicating that the initial balance was invalid. The class should provide three member functions. Member function credit should add an amount to the current balance. Member function debit should withdraw money from the Account and should ensure that the debit amount does not exceed the Account's balance. If it does, the balance should be left unchanged and the function should print a message indicating "Debit amount exceeded account balance." Member function getBalance should return the current balance. Create a program that

creates two Account objects and tests the member functions of class Account.

4. (Employee Class) Create a class called Employee that includes three pieces of information as data members a first name (type string), a last name (type string) and a monthly salary (type int). Your class should have a constructor that initializes the three data members. Provide a set and a get function for each data member. If the monthly salary is not positive, set it to 0. Write a test program that demonstrates class Employee's capabilities. Create two Employee objects and display each object's yearly salary. Then give each Employee a 10 percent raise and display each Employee's yearly salary again.

5. (Date Class) Create a class called Date that includes three pieces of information as data members a month (type int), a day (type int) and a year (type int). Your class should have a constructor with three parameters that uses the parameters to initialize the three data members. For the purpose of this exercise, assume that the values provided for the year and day are correct, but ensure that the month value is in the range 1-12; if it is not, set the month to 1. Provide a set and a get function for each data member. Provide a member function displayDate that displays the month, day and year separated by forward slashes (/). Write a test program that demonstrates class Date's capabilities.

6. Create a class called Kitten that has three fields: String name, String owner, and int age. Create a constructor for Kitten that takes a String name and owner for the Kitten and uses them for initialization. Have the age for a Kitten start at 0; Implement a method for age which return a Kitten’s age. Implement a method called haveBirthday that does not return anything and simply increases a Kitten’s age by one. Finally, write a method called toString that returns a string of the form: ”<Kitten name> is <age> and belongs to <Owner name>” e.g. ”Bob is 87 and belongs to Gregor Samsa”. Create an object which initialize three data members and print out the string above

7. Define a class called Triangle with three integer data members a, b, and c as the lengths of its three edges. This class should have the following methods:
   (a)a constructor with 3 parameters representing the 3 edges
   (b)a method isTriangle() which returns true if the 3 edges are all

positive and they satisfy the triangle inequality where a+b > c, a+c > b, b+c > a.
(c)a method getArea() which returns the area of triangle. NB: Area=1/2(a*b)
The signature of these methods are given below:

```
public Triangle(int newa, int newb, int newc)
public boolean isTriangle()
public double getArea()
```

Note: getAngle() should return zero if the triangle is not really a triangle.

8. Create class SavingsAccount. Usea static variable annualInterestRate to store the annual interest rate for all account holders. Each object of the class contains a private instance variable savingsBalance indicating the amount the saver currently has ondeposit. Provide method calculateMonthlyInterest to calculate the monthly interest by multiplying the savingsBalance by annualInterestRate divided by 12 this interest should be added to savingsBalance. Provide a static method modifyInterestRate that sets the annualInterestRate to a new value. Write a program to test class SavingsAccount. Instantiate two savingsAccount objects, saver1 and saver2, with balances of $2000.00 and $3000.00, respectively. Set annualInterestRate to 4%, then calculate the monthly interest and print the new balances for both savers. Then set the annualInterestRate to 5%, calculate the next month’s interest and print the new balances for both savers.
9. Create a class called Book to represent a book. A Book should include four pieces of information as instance variables-a book name, an ISBN number, an author name and a publisher. Your class should have a constructor that initializes the four instance variables. Provide a mutator method and accessor method (query method) for each instance variable. Inaddition, provide a method named getBookInfo that returns the description of the book as a String (the description should include all the information about the book). You should use this keyword in member methods and constructor. Write a test application named BookTest to create an array of object for 30 elements for class Book to demonstrate the class Book's capabilities.

10. (Complex Class) Create a class called Complex for performing arithmetic with complex numbers. Write a program to test your class.
    Complex numbers have the form realPart + imaginaryPart * I
    where i is
    Use double variables to represent the private data of the class. Provide a constructor that enables an object of this class to be

initialized when it is declared. The constructor should contain default values in case no initializers are provided. Provide public member functions that perform the following tasks:

a. Adding two Complex numbers: The real parts are added together and the imaginary parts are added together.
b. Subtracting two Complex numbers: The real part of the right operand is subtracted from the real part of the left operand, and the imaginary part of the right operand is subtracted from the imaginary part of the left operand.
c. Printing Complex numbers in the form (a, b), where a is the real part and b is the imaginary part.

11. (Rational Class) Create a class called Rational for performing arithmetic with fractions. Write a program to test your class.

    Use integer variables to represent the private data of the classthe numerator and the denominator. Provide a constructor that enables an object of this class to be initialized when it is declared. The constructor should contain default values in case no initializers are provided and should store the fraction in reduced form. For example, the fraction
    would be stored in the object as 1 in the numerator and 2 in the denominator. Provide public member functions that perform each of the following tasks:

    a. Adding two Rational numbers. The result should be stored in reduced form.
    b. Subtracting two Rational numbers. The result should be stored in reduced form.
    c. Multiplying two Rational numbers. The result should be stored in reduced form.
    d. Dividing two Rational numbers. The result should be stored in reduced form.
    e. Printing Rational numbers in the form a/b, where a is the numerator and b is the denominator.
    f. Printing Rational numbers in floating-point format.

12. (HugeInteger Class) Create a class HugeInteger that uses a 40-element array of digits to store integers as large as 40 digits each. Provide member functions input, output, add and substract. For comparing HugeInteger objects, provide functions isEqualTo, isNotEqualTo, isGreaterThan, isLessThan, isGreaterThanOrEqualTo and isLessThanOrEqualToeach of these is a "predicate" 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. If you feel ambitious, provide member functions multiply, divide and modulus.

13. (TicTacToe Class) Create a class TicTacToe that will enable you to write a complete program to play the game of tic-tac-toe. The class contains

as private data a 3-by-3 two-dimensional array of integers. The constructor should initialize the empty board to all zeros. Allow two human players. Wherever the first player moves, place a 1 in the specified square. Place a 2 wherever the second player moves. Each move must be to an empty square. After each move, determine whether the game has been won or is a draw. If you feel ambitious, modify your program so that the computer makes the moves for one of the players. Also, allow the player to specify whether he or she wants to go first or second. If you feel exceptionally ambitious, develop a program that will play three-dimensional tic-tac-toe on a 4-by-4-by-4 board. [Caution: This is an extremely challenging project that could take many weeks of effort!]