Practice Examples for Lab: Set 15

- 1

Define a class for storing polynomials. Assume that all your polynomials will have degree at most 100. Write a member function `value` which takes a polynomial and a real number as arguments and evaluates the polynomial at the given real number. Overload the `+,*,-` operators so that they return the sum, product and difference of polynomials. Also define a member function `read` which reads in a polynomial from the keyboard. It should ask for the degree d of the polynomial, check that $d \leq 100$, and then proceed to read in the first $d+1$ coefficients from the keyboard. Define a `print` member function which causes the polynomial to be printed. Make sure that you only print $d+1$ coefficients if the actual degree is d. Carefully decide which members will be private and which will be public. Overload the `>>`, `<<` operators so that the polynomial can be read or printed using them.

Practice Examples for Lab: Set 15

- 2

Define a class for storing complex numbers. Provide 0, 1, 2 argument constructors which respectively construct the complex number 0, a complex number with imaginary part 0 and real part as specified by the argument, and a complex number with real and imaginary parts as specified by the arguments. Overload the arithmetic operators to implement complex arithmetic.

- 3

  Implement the following operators for the Ratio class:
  addition, subtraction, multiplication, division, <, >, ==, =

- 4

  Implement the following operators for the Point class:
  =, << (for output), ==, !=, +, -, * (inner product)

Practice Examples for Lab: Set 15

- 5

Define the operator `>>` for the class `V3`. This should enable you to write `cin >> v;` where `v` is of type `V3`. When this is executed, the user will type in 3 floating point numbers which will get placed in `v`.

- 6

Implement a `Matrix` class for 2-by-2 matrices:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Include a default constructor, a copy constructor, an `inverse()` function that returns the inverse of the matrix, a `det()` function that returns the determinant of the matrix, a Boolean function `isSingular()` that returns 1 or 0 according to whether the determinant is zero, and a `print()` function.

Practice Examples for Lab: Set 15

- 7

Implement a `Circle` class. Each object of this class will represent a circle, storing its radius and the x and y coordinates of its center as `floats`. Include a default constructor, access functions, an `area()` function, and a `circumference()` function.

- 8

Implement a `Point` class for two-dimensional points (x, y). Include a default constructor, a copy constructor, a `negate()` function to transform the point into its negative, a `norm()` function to return the point's distance from the origin (0,0), and a `print()` function.

- 9

Implement a `Time` class. Each object of this class will represent a specific time of day, storing the hours, minutes, and seconds as integers. Include a constructor, access functions, a function `advance(int h, int m, int s)` to advance the current time of an existing object, a function `reset(int h, int m, int s)` to reset the current time of an existing object, and a `print()` function.