**CMPE 50 – Fall 2018, Tarng**

**Lab #8– Classes and Dynamic Arrays**

**Instructions**: Before you leave the lab, you should submit your answers through Canvas->Assignment->Lab8->Submission. Please submit your answers (.h and .cpp files, and output files with appropriate documentation/comments) even if you couldn’t complete/run them.

**Exercise 1:**

(Based on Programming Project 11.9)

Using dynamic array, implement a polynomial class with polynomial addition subtraction, and multiplication. The partial solution has been posted to Canvas in Files->Lab Assignment. Please extend the solution to complete the following operations:

* polynomial + polynomial
* constant + polynomial
* polynomial + constant
* polynomial – polynomial
* constant – polynomial
* polynomial – constant
* polynomial \* polynomial
* constant \* polynomial
* polynomial \* constant

Supply functions to assign and extract coefficients, indexed by the exponents (degrees).

Supply a function to evaluate the polynomial given a value of type double.

**Exercise 2:**

(Based on Programming Project 11.6)

You need to write an interface file and an implementation file for the class definition. Also you need to write a test driver program for testing.

Define a class for complex numbers. A complex number is a number of the form

a + b \* i

where, for our purposes, a and b are numbers of type double, and i is a number that represents the quantity √-1 (square root of -1). Represent a complex number as two values of type double. Name the member variables real and imaginary. (The variable for the number that is multiplied by i is the one called imaginary.) Call the class Complex.

Include a constructor with two parameters of type double that can be used to set the member variables of an object to any values. Also include a constructor that has only a single parameter of type double; call this parameter real\_part and define the constructor so that the object will be initialized to real\_part+0\*i. Also include a default constructor that initializes an object to 0 (that is, to 0+0\*i). Overload all of the following operators so that they correctly apply to the type Complex: ==, +, -, \*, >>, and <<. You should write a test program to test your class.

Hint: To add or subtract two complex numbers, you add or subtract the two member variables of type double. The product of two complex numbers is given by the following formula:

(a + b \* i) \* (c + d \* i) == (a \* c – b \* d) + (a \* d + b \* c) \* i

In the interface file, you should define a constant I as follows:

const Complex i(0,1);

This defined constant i will be the same as the i discussed earlier.