2016/05/10

Program Design
Second Mid-Test – 2016

☞ **No Dictionaries (paper or electronic) allowed.**
☞ **No cellular phone (big or small) allowed.**
☞ **No Conversations (loud or murmur) allowed**

I. Matching: choose the correct meanings (30 %)

d **1. Free store or heap**	g **2. new and delete**
c **3. Iterator**	o **4. Memory leak**
b **5. Data abstraction**	e **6. friend function**
n **7. this pointer**	m **8. new and delete**
f 9. **Null** character	a **10. static class variable**
l 11. Inheritance	s **12. Memberwise assignment**
k 13. "Is a" relationship	p **14. Operator overloading**
j 15. Derived class	i 16. Multiple inheritance
h 17. Base class	q **18. Conversion constructor**
t 19. Dangling pointer	r 20. Single-argument constructor

a) Used when only one copy of a variable should be shared by all instances of a class.
b) Describing functionality of a class independent of its implementation.
c) An object that "walks through" a collection.
d) A region of memory for storing objects created at execution time.
e) Defined outside the class's scope, yet has access to private members of the class.
f) String termination character.
g) Operators used for performing dynamic memory allocation and deallocation.
h) Class from which others are derived.
i) Deriving from more than one base class.
j) Class that is created by inheriting from an existing class.
k) The inheritance relationship.
l) New classes are created from existing classes.
m) Operators used for performing dynamic memory allocation and deallocation.
n) An implicit argument to all non-static member-function.
o) Occurs when objects are allocated but never deallocated.
p) Enables C++'s operators to have class objects as operands.
q) A constructor that takes as its argument a reference to an object of the same class as the one in which the constructor is defined.
r) A constructor that transforms its one parameter into an object of the class.
s) The default behavior of the = operator.
t) Problem that may occur when default memberwise copy is used on objects with dynamically allocated memory.

II. Complex numbers have the form: (50%)

realPart + *imaginaryPart* * *i*

where *i* has the value $\sqrt{-1}$

please create a class **Complex;** use **double** type variables to represent the private data *realPart* and *imaginaryPart.*

a) Define a **constructor** that accept two arguments, e.g. **3.2, 7.5.** to initialize the data members by using **member-initializer syntax**.

make this constructor a **default constructor** too by assigning the two data members both to values 1.0. The constructor also prints out a message like:

Complex number (3.2, 7.5) is constructed.

b) Define a **destructor** that prints a message like:

Complex number (3.2, 7.5) is destroyed.

c) Define a copy constructor that creates a complex number object and initializes by using another complex number object.

d) Write a public member function ***Add*** that adds another complex number to *this* complex number object.

e) Overload the + operator to perform the same function as ***Add*** function.

f) Write a public member function ***PrintComplex*** to print a Complex numbers in the form:

realPart + *imaginaryPart***i*

g) Overloads both the << and >> operators (with proper friendship declarations) to output (as in f.) and input two double values for a Complex number.

h) Overload the == and the **!=** operators to allow comparisons of complex numbers. (please use definition of == to define **!=**)

i) Overload the assignment operator = to allow mutual assignments between complex number objects.

j) Overload the ++ and the -- operators for ***pre-*** and ***post-*** operations that adds 1 to and minus 1 from both the ***realPart*** and the ***imaginaryPart***.

III. Define a *Polynomial* class for holding a second order polynomial with at most three terms as:

$7X^2+4X+9$ or $8X^2-0X+7$

use an integer pointer data member **int *coeff** that points to a dynamically allocated **int array of three elements** using **new** operator in the constructors. Each element contains the coefficient of an exponent. For example the three elements of the array would have 7, 4, 9, and for the other one would have 8, 0, 7. (20%)

a) Define a constructor with three arguments that sets the elements of the dynamically allocated array to the three arguments. Make it a default constructor too that sets the elements of the dynamically allocated array to 0s.

b) Define a copy constructor that sets the elements of the dynamically allocated array to the elements of the object argument.

c) Define a destructor that will return the dynamically allocated array to the heap.

d) Overload the **static cast operator** that convert a ***Polynomial*** object to an integer that is the sum of the three coefficients.