50 points total

PROG 113 Midterm Exam

KEEP AS SIMPLE AS POSSIBLE – KEEP EVERYTHING IN ONE FILE

General Instructions: The Exam is an *open book*, *open notes* exam; you may use your textbook and any other notes that you have. You will complete this assignment using the Visual Studio IDE to code and test your solution. You may ***not***, however, talk to anyone about anything during the "open" exam period. You may not receive answers or help from anyone. You will have 4 full days to work on the Exam. When you have finished, you must zip your solution as you do with your project assignments, namely the .cpp source code file and the Release, executable, *.exe, file**.** You do ***not*** need to supply a *Project Summary* document. **Make sure to submit your project vial the Canvas Assignment submission by 11:59 pm on Monday evening, 18 May, 2015.** Please note that submissions will be *locked* 3 minutes after the due date/time. There will be ***no*** exceptions for late submissions! If you are not done before the due date, stop 5 minutes ahead and zip your solution as it exists and submit it before the locked date/time. I will grade incomplete solutions for partial credit.

You ***must*** use the supplied C++ source code file the start your project. Create a Visual Studio project and include the supplied file in your project. You may do this by, 1) Creating a project from the template as you have done with project assignments and then replace the source code file contents with the entire contents of the supplied source code file, or 2) You may remove/delete the template source code file from your project and then *copy* the supplied file to the *Project* folder in your Visual Studio *Solution* folder, then go to the *PROJECT* tab in Visual Studio, select "*Add Existing Item…*" and add the file to your project. Make sure you **add** the file to your project if you chose Option #2.

You will be creating two classes in addition to the Client program in method *main()*. The source code file supplied will contain the "skeleton" definitions of these two classes. You will need to fill-in the details, specifically required data members and method definitions. You will ***not*** need to define separate header (*.h) and method definition (*.cpp) files for each of the two classes you need to define. You will add methods/data members as required to the supplied class definitions and then define the required methods externally following the two class definitions. You are free to add any "helper" methods as you desire to the Class definitions. There is no need, and therefore you should *not*, make changes to any of the supplied Client code in method *main()*, as I will use this Client code to test your final solution. You will be responsible for defining the methods and Operators declared in the two Class definitions!

Class *Mail*:

- The Base class, which defines the basic properties of any type of mail item.
    A. Default Constructor
    B. Initializing Constructor
    C. The type of mail, a C-String description
    D. A C-String, *char* array to store the mail type description, fixed size
    E. Default mail type: First Class Mail
    F. Default cost/oz.: .$ 0.49
    G. The weight of the item (oz)
    H. Default weight: 1 oz.
    I. The cost, per ounce to mail the item, < 0 indicates a "fixed" cost, The absolute value is the fixed cost to mail the item (a *Package*)
    J. Method to RETurn the total cost (accounts for a fixed price item)
    K. Overloaded Assignment Operator (accounts for C-String array!)
    L. Copy Constructor (accounts for C-String array!)
    M. Overloaded Addition (+) Operator: Adds weight (oz) to item
    N. Overloaded Insertion Operator (<<): Description of *Mail* item

Class *Package*:

- Class derived from Base Class *Mail*:
    A. Initializing Constructor
    B. Default type of mail: "Express Mail" (passed to *Mail* Constructor)
    C. Fixed cost: Supplied in the Initializing Constructor (passed to *Mail*)
    D. Weight (oz): Passed to *Mail* Constructor
    E. Adds data member: Guaranteed days for delivery
    F. Copy Constructor
    G. *No* requirement to overload the Assignment Operator
    H. Overloaded Insertion Operator (<<): Description of *Package* item Description of *Mail* plus number of guaranteed delivery days

The expected Console output of the provided Client code, *main()*, is shown on the next page.

```
            PROG-113: Midterm Exam

This Client program will test the various requirements for creating,
copying, adding an "int" value to an object, and assigning one object
to another object. The objects are of two types, 1) A base class,
"Mail", and 2) A derived class (sub-class), "Package".

"mailItem1":
Mail Class: First Class
    Weight: 1 oz.
Total Cost: $0.49

"mailItem2", a copy of "mailItem1":
Mail Class: First Class
    Weight: 1 oz.
Total Cost: $0.49

Adding 5 oz. to weight of "mailItem1":
Mail Class: First Class
    Weight: 6 oz.
Total Cost: $2.94

"mailItem2", AFTER adding weight to "mailItem1":
Mail Class: First Class
    Weight: 1 oz.
Total Cost: $0.49

"package1":
Mail Class: Express Mail
    Weight: 42 oz.
Total Cost: $7.50
  Delivery: 3 days

"package2", a copy of "package1":
Mail Class: Express Mail
    Weight: 42 oz.
Total Cost: $7.50
  Delivery: 3 days

Cost of "mailItem1": 2.94
Cost of "mailItem2": 0.49
Cost of "package1":  7.50

Press any key to continue . . .
```