CSC 2431 Assignment 2
Due: Friday 4/22/16 by 11:59pm

Part 1 – Textbook Exercises

The following textbook exercises are assigned to reinforce the concept of inheritance:
Ch. 11: #12, 14 and Ch. 12: #34, 36

Part 2 – Programming Assignment

Consider the `rectangleType` and `boxType` classes we developed using Malik's definitions from the text. For this assignment, you will create a new derived class *`treasureChest`* from `boxType`. Start with the baseline code on Blackboard for `rectangleType` and `boxType`, and add the following to `boxType`:

- Add a `string pattern` as a private member variable to `boxType` along with a `setPattern` mutator and `getPattern` accessor. This member variable keeps track of the 2-D pattern printed on the surface of a `boxType` object.
- Modify the parameterized constructor for `boxType` to take a parameter to set `pattern`. The default constructor should set `pattern` to an empty `string`.

Next, derive `treasureChest` from `boxType` using ***public inheritance***. `treasureChest` has the following properties:

- Note: this description is deliberately *not* a UML class diagram or similar. **You should sketch an outline in pseudocode for your new class as you read these descriptions (and the client) to ensure you meet all the requirements.** (You don't have to turn your sketch in.)

- A `treasureChest` can store treasure! Treasure in this case will be represented by a dynamic array* of `coin` variables. Use an enumerated type to represent a `coin`, with the following three types and values:
    - gold (value 25)
    - silver (value 10)
    - copper (value 1)

  *We have exhaustively looked at dynamic arrays in CSC 2430. You are welcome to use a `dynArray` to store the `coin`s, or use code from CSC 2430 in your implementation, **but** you must cite your source if not writing from scratch (even if using your own lab solutions!).

- A `treasureChest` has a *maximum* number of `coin`s that can be stored. Keep track of this with a private member variable (an `int`). The default is 0.
    - Create an appropriate accessor to return the maximum size of the chest.

- A `treasureChest` has a *current total* number of *`coin`s* that are stored. Keep track of this with a private member variable (an `int`). The default is 0.
    - Create an appropriate accessor to return the total number of coins in the chest.

- Treasure can be added to the chest via a class method that allocates a new `coin` on the heap and adds it to the dynamic array (assuming there is room in the chest).

- Treasure can be removed from the chest one piece at a time with a method that takes a `coin` as input parameter and removes *one* `coin` of that type from the dynamic array (if any of that type are present).

- The pattern on the `treasureChest` is restricted to two options:
    - "jolly roger" or "East India Company flag"
    - The `treasureChest` constructors should set `pattern` appropriately (the default will be an empty `string`, but make sure to follow the "flag" rule).
- Make sure to create an appropriate destructor. Ensure all dynamically allocated memory is freed.

What to turn in:

1. Your handwritten or typed answers to the textbook exercises.
2. Your header (.h) and source (.cpp) files for `rectangleType`, `boxType` and `treasureChest`.
    a. You may modify `rectangleType` and `boxType` as needed. **But** you must keep the private members as private.
3. A copy of the client (**it should be unmodified,** but this makes it easier to test your code).
4. Your output after running the test code in the client.
5. All documents should be uploaded to Blackboard following the assignment submission instructions (linked on Blackboard). The textbook exercises may be handwritten and turned in during class or under the instructor's office door by the due date.