Concordia University
Department of Computer Science
and Software Engineering

Advanced program design with C++
COMP 345 --- Fall 2014

Individual assignment #2

Deadline:	Monday, November 3rd, 2014
Evaluation:	5% of final mark
Late submission:	not accepted
Teams:	this is an individual assignment

Problem statement

This is an individual assignment. It is divided into three distinct parts. Each individual student is expected to select one of these parts as his/her assignment. Each part is about the development of a part of the topic presented as the team project. Even though it is about the development of a part of your team project, each assignment has to be developed independently of the others and is not to be presented as an integrated part of the team project. Each member of your team is free to choose to do any part, and is expected to follow a different design approach than the other team members that have selected the same assignment topic. Note that the following descriptions describe the baseline of the assignment, and are related to the project description (see the course web page for a full description of the team project).

Part 1: Tower observer

Implement an Observer pattern for the Tower class as implemented in assignment 1. Make the Tower class an Observable class, then implement a Tower Observer class that displays the tower's view. You may use whatever library or solution you see fit to display the character's view (GUI or not). Provide a driver class that creates a tower, plugs a Character Observer to it, then changes some values in the tower, triggering the re-display of the tower view each time a value is changed in the tower using the observer pattern mechanism.

Part 2: Map observer

Implement an Observer pattern for the Map class as implemented in assignment 1. Make the Map class an Observable class, then implement a Map Observer class that displays the map's view. You may use whatever library or solution you see fit to display the map's view (GUI or not). Provide a driver class that creates a map, plugs a Map Observer to it, then changes some values in the map (e.g. moving a critter on the map), triggering the re-display of the map view each time a value is changed in the map using the observer pattern mechanism.

Part 3: Critter observer

Implement an Observer pattern for the Critter class as implemented in assignment 1. Make the Critter class an Observable class, then implement a Critter Observer class that displays the item container's view. You may use whatever library or solution you see fit to display the map's view (GUI or not). Provide a driver class that creates an item container, plugs a Critter Observer to it, then simulate a hit on the critter reducing its hit points, triggering the re-display of the critter view each time hit points are decreasing using the observer pattern mechanism.

Assignment submission requirements and procedure

You are expected to submit a group of C++ files implementing a solution to one of the problems stated above (Part 1, 2 or 3). Your code must include a *driver* that allows the marker to compile and execute your code on a standard lab computer. The driver should simply create a tower, map, or critter object that it is defined and behaves following the tower defense game specifications, and that the Observer pattern is correctly implemented as stated above. The use of test cases is not mandatory but encouraged. Along with your submitted code, you have to explain your design. This can be, for example, as doxygen-generated documentation, regular code comments, or a simple diagram. The focus of this course being the coding aspect of software development, you are discouraged to submit extensive “paper” design documentation.

You have to submit your assignment before midnight on the due date using the ENCS Electronic Assignment Submission system under the category “programming assignment 2”. Late assignments are not accepted. The file submitted must be a .zip file containing all your code. You are allowed to use any C++ programming environment as long as they are available in the labs. No matter what programming environment you are using, you are responsible to give proper compilation and usage instructions to the marker in a README file to be included in the zip file.

Evaluation Criteria

Solution:	
Clarity and correctness of statement of game rules involved:	5 pts
Compliance of solution with stated problem:	20 pts
Simplicity and appropriateness of the solution:	5 pts
Clarity of design description:	5 pts
Programming style:	
Code readability: naming conventions, clarity:	5 pts
Coding style: `.h` and `.cpp` files, use of comments:	5 pts
Relevance of driver and presented results:	5 pts
Total	**50 pts**