CSC 260 Fall 2012

Rubric for Assignment 4 – Abstraction and Aggregation
Part I Due: October 16, 2012 by midnight
Part II Due: October 26, 2012 by midnight

Student Name: _____ **Grade:** _____/60

	Levels of Achievement			
Criteria	**Exceeds Expectations (4)**	**Meets Minimum Expectations (3)**	**Below Expectations (0)**	**Score**
Compiling and Execution	Compiles without errors or warnings. There are no execution errors.		There are compile errors or warnings or many execution errors.	
Note: Assignment is **Incomplete** if the program does not compile without errors. It must be re-submitted after fixing errors.				
Class Diagram	Provides a detailed class diagram including all classes and their relationships in correct UML notation.	Provides a minimal class diagram in mostly correct UML notation.	Provides a trivial or no class diagram, or uses incorrect UML notation.	
Statechart for System	Provides detailed statechart for system behavior in correct UML notation.	Provides minimal diagram, or only for parts of the system in correct UML notation.	Provides a trivial or no statechart, or uses incorrect UML notation.	
Statechart for Operators	Provides detailed diagram with elegant algorithms in correct UML notation.	Provides minimal diagram or inelegant algorithms in mostly correct UML notation.	Provides a trivial or no statechart, or uses incorrect UML notation.	
Test Case Design	Provides detailed test case design that will demonstrate all functionality and exception handling.	Provides minimal test case design that will demonstrate some functionality and exception handling.	Provides a trivial or no test case design, or does not demonstrate most functionality or exception handling.	
Functionality	Provides all functionality as per specs, and does not implement user i/o in functions except where required.	Provides most functionality as per specs, and may implement some inappropriate user i/o in functions.	Does not provide most significant functionality, and implements user i/o inappropriately in most functions.	
Problem Solution	Implements correct, elegant and efficient algorithms.	Implements working but inelegant algorithms.	Implements incorrect algorithms that do not meet specifications.	
Overloading Operators	Elegantly handles operator overloading to meet all the requirements.	Trivially handles operator overloading to meet some of the requirements.	Does not implement operator overloading as specified.	
Templates	Elegantly implements template classes to meet all the requirements.	Trivially implements template classes to meet some of the requirements.	Does not implement template classes as specified.	
Use of OOP and C++	Implements excellent encapsulation and information hiding with appropriate level of modularity. Makes excellent use of C++ constructs and parameter passing and does not use global variables.	There is some violation of encapsulation and information hiding, with inadequate modularity. There is some inappropriate use of C++ constructs, parameter passing or global variables.	Does not satisfy encapsulation or information hiding and is not modular. There is considerable inappropriate use of C++ constructs and global variables.	

CSC 260 Fall 2012

	Levels of Achievement			
Criteria	**Exceeds Expectations (4)**	**Meets Minimum Expectations (3)**	**Below Expectations (0)**	**Score**
Programming Practices	Indents and formats code well, with consistent placement of braces and tab spacing, to improve readability.	Indents and formats code for most part, with some inconsistent placement of braces and tab spacing that hinders readability.	Does not indent or format code so that readability is severely hindered.	
	Organizes classes and methods well, and uses good and consistent naming convention to promote reuse and ease of maintenance.	Provides mostly well-organized classes and methods, and uses mostly consistent naming convention that promote reuse and ease of maintenance.	Does not organize classes or methods, or uses unintuitive naming convention, severely constraining reuse and ease of maintenance.	
Documentation	Provides detailed identification information and clear and detailed comments as appropriate.	Provides minimum identification information and/or minimal comments as appropriate.	Does not provide identification information or appropriate comments.	
Exception Handling	Implements graceful error handling within methods.	Implements minimal error handling or distributes it between methods and driver.	Implements no error handling within methods.	
Testing	Demonstrates extensive testing of all functionality and exception handling guided by the test case design.	Demonstrates some testing of functionality and exception handling, not necessarily guided by the test case design.	Does not adequately demonstrate functionality and exception handling.	
Deliverables	6 points penalty for posting source code on the wiki or team's svn repository.			
Timeliness	6 points penalty for each day late.			

Comments: