Concordia University
Department of Computer Science
and Software Engineering

Advanced Program Design with C++
COMP 345 --- Fall 2015

Project Final Build Grading

1. First Incremental Code Build Description

You must deliver an operational version demonstrating the full capacity of your system. This is about demonstrating that the code build is effectively aimed at solving specific project problems and completely implementing specific system features. The code build must not be just separated portions of the final project, but a fully operationally integrated software that can be demonstrated by its operational usage.

The presentation should be organized as follows:

1. Brief presentation of the Problem Analysis, Design, and Use of Engineering Tools as listed below under "Graduate attributes—skills**"**
2. Demonstration of the functional requirements as listed below under "Functional Requirements".

You are graded according to how effectively you can demonstrate that the features are implemented. If you cannot really demonstrate the integrated features through execution, you will have to prove that the features are implemented by explaining how your code implements the features and what are the expected integration problems, in which case you may lose some marks, even if your explanations are satisfactory.

During your presentation, you have to demonstrate that you are well prepared for the presentation, and that you can easily provide clear explanations as questions are asked about your understanding of the problem being solved, the structure and functioning of your code, as well as your use of tools.

2. Team Identification

Team	Evaluator	Signature	Date	Time

3. Grading

Functional Requirements			**35**
Map creation and editing			**7**
User-driven creation of map elements, such as country, continent, and connectivity between countries.			1
Saving a map to a file exactly as edited			1
Loading a map from an existing file, then editing the map			1
Use of the Adapter pattern to save/load from two different map file formats (see individual assignment 3).			2
Verification of map correctness upon saving/loading (at least 3 types of incorrect maps, including verification that the map is a connected graph, and that each continent is a connected graph)			2
Game driver			**4**
Implementation of a round-robin loop for players' turns/phases, identification of a winner and end of game, defeated player is removed from the round-robin loop.			2
Save/Load a game in progress using a Builder pattern (see individual assignment 3).			2
Game display			**8**
Player Observer that displays relevant information about a player (see individual assignment 2).			2
Map Observer that displays relevant information about the map (see individual assignment 2).			2
Game statistics Observer/Decorator that displays user-selected game statistics (see individual assignment 3).			2
Game log Observer/Decorator that displays logging of user-selected players/phases (see individual assignment 3).			2
Startup phase			**5**
Game log and game statistics Observers are initially set to show user-selected parts (see individual assignment 3) using the Observer and Decorator patterns.			1
User selection of a previously user-saved map, then load the map.			1
User chooses the number of players, all countries are randomly assigned.			1
Types of players are initially assigned and implemented using a Strategy pattern (see individual assignment 2). Types of players can be changed at any time during play.			2
Reinforcement phase			**4**
Calculation of correct number of reinforcement armies and placement of armies on the map.			1
Correct implementation of cards and their additional reinforcements.			3
Attack phase			**6**
Correct identification of valid attacking/attacked country.			1
Correct implementation of an attack using the Risk battle model.			3
Correct implementation of the post-battle movement after a victory.			1
A victory gives a card, eliminating a player transfers all the defeated player's cards.			1
Fortification phase			**1**
Implementation of a single valid move according to the Risk rules.			1
Graduate attributes—skills			**15**
Knowledge-base	Indicator 1.3: Knowledge-base in a specific domain: demonstrated knowledge of programming principles used in the implementation.		1
Design	Indicator 4.1: Problem identification and information gathering: knowledge and correct understanding of the functional requirements and the game rules.		2
	Indicator 4.3: Architectural and detailed design: Rationale for overall project architectural structure. Demonstration/explanation of the correct use of three different design patterns such as those implemented in the individual assignments.		3
	Indicator 4.4: Implementation and validation: Correct use of C++ features leading to stable execution that has been properly tested in various situations.		2
Use of tools	Indicator 5.1: Ability to use appropriate tools, techniques and resources: proficient use of particular tools (C++ language, libraries, project management tools, etc.) for the implementation.		2
	Indicator 5.2: Ability to select appropriate tools, techniques, and resources: justified adoption of tools in the project (e.g. compiler, IDE, libraries, project management tools, etc).		1
Communication	Indicator 7.3: Documentation: Code readability: layout, naming. Consistent use of comments.		2
	Indicator 7.4: Oral presentation: Structure and demonstrated preparation of presentation, using appropriate presentation techniques. Demonstrated knowledge of code base/clarity of explanations.		2
Total			**50**