Concordia University

Department of Computer Science and Software Engineering

Advanced Programming Practices SOEN 6441 --- Winter 2019

Project Build 1 Grading

Deadline:	Submission of the project: March 5, 2019, 23:55pm Presentation: March 6, 7, and 8, 2019 during lab hour, time schedule will be posted soon.
Evaluation:	Intermediate delivery 1: 10%
Late submission: **Teams:**	not accepted the project is a team assignment

INSTRUCTIONS FOR INCREMENTAL CODE BUILD PRESENTATION

You must deliver an operational version demonstrating a subset of the capacity of your system. This is about demonstrating that the code build is effectively aimed at solving specific project problems or completely implementing specific system features. The code build must not be just a "portion of the final project", but rather be something useful with a purpose on its own, that can be demonstrated by its operational usage.

The presentation should be organized as follows for **10 minutes**:

1. Brief presentation of the goal of the build.
2. Brief presentation of the architectural design of your project.
3. Demonstration of the functional requirements as listed on the following grading sheet.
4. Demonstration of the use of tools as listed on the following grading sheet.

You are graded according to how effectively you can demonstrate that the features are implemented. If you cannot really demonstrate the features through execution, you will have to prove that the features are implemented by explaining how your code implements the features, in which case you will get only partial marks.

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 the functioning of your code, or your required usage of the tools/techniques.

On March 5, 2019, 23:59pm, you have to submit your current project code base to the Electronic Assignment Submission System under “project 1”.

GRADING

Presentation		**5**
Effectiveness, structure and demonstrated preparation of the presentation		2
Fluid exposition of knowledge of code base/clarity of explanations		3
Functional Requirements		**30**
Map editor		**12**
User-driven creation of map elements, such as country, continent, and connectivity between countries.		4
Saving a map to a text file exactly as edited (using the "conquest" game map format).		3
Loading a map from an existing "conquest" map file, then editing the map, or create a new map from scratch.		3
Verification of map correctness upon loading and before saving (at least 3 types of incorrect maps).		2
Game Play		**18**
Implementation of a game driver implementing the game phases according to the Risk rules.		2
Start-up phase		**8**
Game starts by user selection of a user-saved map file.		1
Map is loaded as a connected graph, which is rendered effectively to the user to enable efficient play.		3
User chooses the number of players, then all countries are randomly assigned to players.		1
Players are allocated a number of initial armies, depending on the number of players.		1
In round-robin fashion, the players place their given armies one by one on their own countries.		2
Reinforcement phase		**5**
Calculation of correct number of reinforcement armies according to the Risk rules.		2
Player place all reinforcement armies on the map.		3
Fortification phase		**3**
Implementation of a valid fortification move according to the Risk rules.		3
Programming process		**15**
Architectural design—short document including an architectural design diagram. Short but complete and clear description of the design, which should break down the system into cohesive modules. The architectural design should be reflected in the implementation of well-separated modules and/or folders.		3
Software versioning repository—well-populated history with dozens of commits, distributed evenly among team members, as well as evenly distributed over the time allocated to the build. A tagged version should have been created for build 1.		3
API documentation—completed for all files, all classes and all methods.		3
Unit testing framework—at least 10 relevant test cases testing the most important aspects of the code. Must include tests for: (1) map validation; (2) calculation of number of reinforcement armies; (3) reading an invalid map file.		3
Coding standards—documented description of coding standard used. Consistent and proper use of code layout, naming conventions and comments, absence of "commented out" code.		3
Total		**50**

Notes