**Concordia University**
**Department of Computer Science**
**and Software Engineering**

**Advanced Program Design with C++**
**COMP 345 --- Fall 2016**

# Project Intermediate 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 Design, and Use of 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** |
| **Item creation/editing** | | | **3** |
| User-driven creation of validated items (armor, ring, helmet, boots, belt, sword, shield) | | | 2 |
| Loading/saving an item to a file as edited | | | 1 |
| **Map creation/editing** | | | **10** |
| User-driven creation of a validated map by placing elements such as entry/exit door, walls, chest, and characters (friendly or hostile). | | | 4 |
| Saving a map to a file as edited. | | | 2 |
| Loading a map from an existing file, then editing the map. | | | 2 |
| Verification of map correctness before saving (at least 3 types of incorrect maps). | | | 2 |
| **Character creation/editing** | | | **10** |
| User-driven creation/editing/validation of a *fighter* character following the d20 game rules: level, ability scores and modifiers, hit points, armor class, attack bonus, damage bonus, multiple attacks, item enhancement. | | | 3 |
| Inventory pane, including worn items slots (one of each: armor, ring, helmet, boots, belt, sword, shield) and backpack (item container). Equip /unequip items, i.e. move items between worn item slots and the backpack. | | | 4 |
| Saving/loading a character to/from a file | | | 3 |
| **Play** | | | **12** |
| Selecting a map and a character from a list of saved ones | | | 2 |
| Adapting the map elements (opponents, treasure) to the level of the character upon entry | | | 2 |
| Starting the game by having the player character placed on the starting point | | | 1 |
| Moving the character, square by square on the map | | | 2 |
| Toggling a view of character information (player or opponents) and chest content during play. | | | 2 |
| Ending the game by having the character stepping on the exit point and going up a level | | | 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 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. Use of Doxygen code documentation. | | 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** |