Concordia University
Department of Computer Science
and Software Engineering

Advanced program design with C++
COMP 345 --- Fall 2013

Individual assignment #1

Deadline:	Friday, October 18th, 2013
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, or include the implementation of one another's aspects. 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: Character

Implement a group of C++ classes that allow the generation of player characters following the d20 game rules. The baseline of this assignment is to create characters belonging to the *fighter* class. The character constructor must automatically generate the following, based on the level (provided at creation time) and class of the character: (1) ability scores (generated randomly) and ability modifiers, (2) hit points (based on constitution modifier and level), (3) armor class (based on dexterity modifier), (4) attack bonus (based level and strength/dexterity modifiers), (5) damage bonus (based on strength modifier). The character must be able to wear items of the following kinds and restricted to the following maximum number of items worn for each: 1 helmet, 2 rings, 1 weapon, 1 shield, 1 armor, 1 belt, 1 boots.

Part 2: Map

Implement a group of C++ classes that allow the generation of custom maps. This must allow for the creation of custom maps of variable size to be determined prior to the creation of the map. As stated in the project description, it is advised (for simplicity) that you design the map as a grid, where each grid cell is either (1) an empty cell where a character can move or (2) a wall where a character cannot move, or (3) an occupied cell containing a character, opponent, chest, begin and end cell, or whatever else your game rules might include. The class should be designed to allow the creation of a blank map given breadth and width, and provide member functions to set any cell to anything it might eventually contain as stated above. Your class should have a method to verify the validity of a map, which verifies that there is at least one clear path between the mandatory begin and end cell.

Part 3: Items and item container

Implement a group of C++ classes that implement an item container. Items can be of type helmet, ring, weapon, shield, armor, belt and boots. Items may be enchanted with a +1 to +5 enchantment bonus that upon wearing will eventually influence one of the character's statistics (Strength, Dexterity, Constitution, Intelligence, Wisdom or Charisma), armor class, or attack and damage bonus. Different types of items should provide only with a certain

possibility of enhancement types as per the table below. Each item must have a member function to return the enhancement it bears (enhancement type and bonus) in order to apply it to the character that wears it The item container may contain any number of items and must provide member functions to get a specific item from it, or put a new item into it. Eventually the following item containers will be needed in the game: character backpack, character worn items, treasure chest and merchant.

Item	May increase either
Helmet	Intelligence, Wisdom, Armor class
Armor	Armor class
Shield	Armor class
Ring	Armor class, Strength, Constitution, Wisdom, Charisma
Belt	Constitution, Strength
Boots	Armor class, Dexterity
Weapon	Attack bonus, Damage bonus

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 character, map, or item container object and somehow demonstrate that the character, map or item container was created following the above-mentioned specifications, as well as following the applicable d20 game rules. The use of unit testing such as cppUnit is not mandatory but encouraged. Along with your submitted code, you have to explain your analysis and design. Briefly explain the game rules involved in the creation of you assignment, citing external sources for specific game rules. Briefly describe the design you adopted as a solution. The design description can be backed-up, for example, by 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 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 1”. 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 usable 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**