Concordia University
Department of Computer Science
and Software Engineering

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

Individual assignment #3

Deadline:	Friday, November 22nd, 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. 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 builder

Implement a Builder pattern for the Character class to create characters (player character or enemy character) of various levels (fighter class only), and enabling various types of fighter style to be chosen.

Ability scores generation method: Any character has the same 6 ability scores (Strength, Intelligence, Dexterity, Constitution, Wisdom, Charisma). Upon creation of the character, the values associated with each ability score is randomly determined. For the generation of ability scores, for each ability score, roll 4d6 and selects the 3 highest dice values. After all 6 scores have been generated, they are assigned to an ability depending on the type of fighter that this character is: (1) a *"bully"* uses brute strength to destroy his enemies, (2) a *"nimble"* favors dexterity and better armor class to evade blows, (3) a *"tank"* favors survival by more hit points through a high constitution score. Create one Concrete Builder for each of these three types of fighter.

Type of fighter	Ability scores in decreasing order of importance
Bully	Strength, Constitution, Dexterity, Intelligence, Charisma, Wisdom
Nimble	Dexterity, Constitution, Strength, Intelligence, Charisma, Wisdom
Tank	Constitution, Dexterity, Strength, Intelligence, Charisma, Wisdom

Level-dependent characteristics: As a character goes up levels, the following are increasing: (1) his hit points go up by (1d10+constitution modifier), (2) his attack bonus goes up by one, and his number of attacks per round increase by one every five levels, according to the following table:

level	1 att/round
1	+1
2	+2
3	+3
4	+4
5	+5

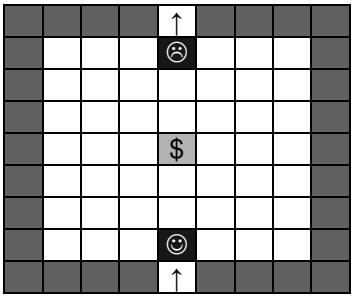
level	2 att/round
6	+6/+1
7	+7/+2
8	+8/+3
9	+9/+4
10	+10/+5

level	3 att/round
11	+11/+6/+1
12	+12/+7/+2
13	+13/+8/+3
14	+14/+9/+4
15	+15/+10/+5

level	4 att/round
16	+16/+11/+6/+1
17	+17/+12/+7/+2
18	+18/+13/+8/+3
19	+19/+14/+9/+4
20	+20/+15/+10/+5

Part 2: Map builder

Implement a Builder pattern for the Map class. One of your Concrete Builders should allow to create an “arena” map. The arena map is a 9X9 map surrounded by walls with the entry point in the middle of the top row, the exit point in the middle of the bottom row, a chest in the middle of the map, and an opposing character blocking the way to the exit point (see figure below). The arena map has a “level” whose value will determine the number and magical strength of the items found in the chest (see “chest builder” below), as well as the level of the opposing character to be defeated in order to exit the map.



Part 3: Chest builder

Implement a Builder pattern for the Chest class. A chest contains magic items usable by a character. Upon opening the chest, the character can select items in the chest and put them in his own inventory. Items can be of the following types: helmet, armor, shield, bracers, ring, belt, boots, sword, bow. Each item has a +1 to +5 modifier that can affect the character in different ways as listed below. Create a Concrete Builder that creates a chest with a random number of randomly-generated items and another Concrete Builder that creates a chest that contains items whose number and strength varies according to the level of the character who opens it.

Item	May increase either
Helmet	Intelligence, Wisdom, Armor class
Armor	Armor class
Shield	Armor class
Bracers	Armor class, Strength
Ring	Armor class, Strength, Constitution, Wisdom, Charisma
Belt	Constitution, Strength
Boots	Armor class, Dexterity
Sword, Bow	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 use the Builder pattern to create character/map/chest objects and somehow demonstrate that the code conforms to the above-mentioned specifications, as well as following the applicable d20 game rules, and that the Builder pattern is correctly implemented. 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 3". 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 available 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**