Concordia University

Department of Computer Science and Software Engineering
Advanced program design with C++

COMP 345 --- F2018 (Section N)
Assignment #1

Deadline: Oct. 9 2018 by 11:55PM (**strict**)

Type: Individual Assignment

Evaluation: 8% of the final mark

Submission: Electronic Submission through your Moodle course homepage. Create an appropriate zip archive including the entire C++ *source* files and all related deliverables.

Problem statement

This is a team assignment. It is divided into five distinct parts. 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 is to be developed/presented/tested separately. The description of each part portrays what are the features that the part should implement, and what you should demonstrate. Note that the following descriptions describe the baseline of the assignment, and are related to the project description [1]. See the course web page for a full description of the team project, as well as links to the details of the game rules to be implemented.

Part 1: Maps
Provide a group of C++ classes that implement a map for the *KING OF NEW YORK* game. The map class must be implemented as a connected graph, where each node represents a zone/ region. Edges between nodes represent adjacency between zones. Each zone can have any number of adjacent zones. Each zone is owned by a player. The map class can be used to represent any map configuration (i.e. not only the "*KING OF NEW YORK*" map). You must deliver a driver that creates a map and demonstrates that the map class implements the following verifications: 1) the map is a connected graph, 2) regions are connected subgraphs and 3) each region is a node. The driver must provide test cases for various valid/invalid maps.

Part 2: Map loader
Provide a group of C++ classes that reads and loads a map file in the .map text file format as found in the "*KING OF NEW YORK*" resource, available online. The map loader must be able to read any such map. The map loader should store the map as a graph data structure (see Part 1). The map loader should be able to read any text file (even ones that do not constitute a valid map). You must deliver a driver that reads various files and successfully creates a map object for valid map files, and rejects invalid map files.

Part 3: Dice Rolling
Provide a group of C++ classes that implement a dice rolling facility to be used during the player's turn. The dice rolling facility should enable the player object to be able to roll up to three times. For the first roll, the player roll the 6 black dice, and for the second and third rolls (both optional), the player can reroll any or all of the dice (even ones that he/she chooses to keep on a previous roll).
Each player will have his own dice rolling object, and each dice rolling object must keep track of the its description symbol value(Energy, Attack, Destruction, Heal, Celebrity, Ouch!) that has been rolled. You must deliver a driver that creates at least 2 dice rolling facility objects, tests that 1) one can request from 1 to 6 dice being rolled, 2) that the container returns the right symbol of values, 3) that no value outside of

the 1-6 dice symbols range is ever returned, 4) that there is an equal share of 1-6 values returned and 5) that every dice rolling facility object maintains all individual values rolled up to now.

Part 4: Player
Provide a group of C++ classes that implement a *KING OF NEW YORK* player using the following design: A player chooses a monster card, and (optional) takes its figure (see Part 5). A player owns a specific zone/region (see Part 1). A player owns the different game tokens; cards (see Part 5). A player has his own dice rolling facility object (see Part 3). A player must implement the following methods, which are eventually going to get called by the game driver: `RollDice()`, `ResolveDice()`, `Move()` and `BuyCards`. You must deliver a driver that creates player objects and demonstrates that the player objects indeed have the above-mentioned features.

Part 5: Cards deck/Monster cards/Tokens/cubes/unit tiles
Provide a group of C++ classes that implement a deck of *KING OF NEW YORK* 64 cards together with the game remaining pieces: the 46 tokens (Web, Jinx, Souvenir, Carapace); the 6 Monster cards; 6 card board figures (optional), enemy cubes, and 45 building/unit tiles.
The deck object is composed of the 64 game cards. Each card has a description (the cost, how to play, effect). Both the deck cards and unit tiles have a `Shuffle()` method that allows a player to shuffle the cards or building/unit tiles (see the game rules for details). You must deliver a driver that creates a deck of the game cards, monster cards, and the above listed game components. The driver must also include the methods that set up the game pieces.

Assignment submission requirements and procedure
You are expected to submit a group of C++ files implementing a solution to all the problems stated above (Part 1, 2, 3, 4, and 5). Your code must include a *driver* (i.e. a main function) for each part that allows the marker to observe the execution of each part during the lab demonstration. Each driver should simply create the components described above and demonstrate that they behave as mentioned above.

Along with your submitted code, you have to explain your design. This can be, for example, as Doxygen [2] generated documentation, regular code comments, or a simple diagram. The use of test cases is not mandatory but encouraged. You are also responsible to give proper compilation and usage instructions in a README file to be included in the zip file.
Important Note: A demo for about 10 minutes will take place with the marker. The demo times will be determined and announced by the markers, and students must reserve (arrange directly with the markers) a particular time slot for the demo. No demo means a zero mark for your assignment.

You have to submit your assignment before midnight on the due date using Moodle under *A#1_SubmissionBox*. 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 you can demonstrate your assignment in the labs.

Evaluation Criteria

Knowledge/correctness of game rules:	2 pts (indicator 4.1)
Compliance of solution with stated problem (see description above):	12 pts (indicator 4.4)
Modularity/simplicity/clarity of the solution:	2 pts (indicator 4.3)
Proper use of language/tools/libraries:	2 pts (indicator 5.1)
Code readability: naming conventions, clarity of code, use of comments:	2 pts (indicator 7.3)
	Total 20 pts (indicator 6.4)

Reference

[1] *KING OF NEW YORK*, Richard Garfield, &Skaff Elias, http://www.iellogames.com/downloads/KONY_rules.pdf

[2] *Doxygen* - Generate documentation from source code, available at: http://www.stack.nl/~dimitri/doxygen/.