Concordia University

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

COMP 345 --- F2018 (Section N)
Assignment #2

Deadline: November 2nd, 2018 by 11:55PM

Type: team of 4 students max.

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 (max 4 students) assignment. It is divided into six 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 describes 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. See the course web page for a full description of the course term project, as well as links to the details of the game rules to be implemented.

Part 1: Game start

Provide a group of C++ classes that implements a user interaction mechanism to start the game by allowing the player to:

1) Select a map from a list of files as stored in a directory.
2) Select the number of players in the game (2-6 players).

The code should load all the game pieces and use the map loader to load the selected and appropriate map, create all the players, assign dice rolling facilities to the players, create a deck of cards, and assign an empty hand of cards to each player.

You must deliver a driver that demonstrates that 1) different valid maps can be loaded and their validity is verified (i.e. it is a connected graph, etc.), and invalid maps are rejected without the program crashing; 2) the right number of players is created, a deck with the right number of cards is created.

Part 2: Game play: startup phase

Provide a group of C++ classes that implements the startup phase following the rules of the KING OF NEW YORK game. This phase is composed of the following sequence:

1. The order of play of the players in the game is determined (players take turns in clockwise order. In order to see who goes first, each player rolls the 6 black dice and the 2 green dice, and whoever rolls the most Attacks starts the game.
2. Starting with the first player, and going clockwise: Player put his/her Monster in the borough of his/her choice, except Manhattan. There can be no more than 2 Monsters in any borough.

You must deliver a driver that demonstrates that 1) all the game pieces have been set either on the map or on the play area according to the game set up; 2) The order of play of the players in the game is determined.

Part 3: Game play: main game loop

Provide a group of C++ classes that implements the main game loop following the rules of the KING OF NEW YORK game. During the main game loop, proceeding clockwise as set up in the startup phase, every player is given the opportunity to do each of the following action during their turn:

1. Roll the Dice (up to 3 times)
2. Resolve the Dice (mandatory)
3. Move (generally optional, but sometimes mandatory)
4. Buy Cards (optional)
5. End Your Turn

This loop shall continue until the end of a turn, that is a player (monster) has reached 20 victory points and survived or if there is only one player (monster) still in the game.
You must deliver a driver that demonstrates that 1) every player gets turns in a clockwise and that their `rollthedice()`, `resolvethedice()`, `move() and buycards` methods are called 2) the game ends when a player reached 20 victory points and survived or he is the only one player left.

Part 4: Main game loop: roll the dice and resolve the dice

Provide a group of C++ classes that implements roll the dice and resolve the dice, following the official rules the KING OF NEW YORK game.
For the role dice, and on the player's turn, he can roll the dice up to three times. For his first roll, he will roll the 6 black dice (the green dice are only used with particular cards).
For player second and third rolls (both optional), he can reroll any or all of the dice (even ones that he chose to keep on a previous roll).
For resolve the dice, the symbols on the dice after his final roll indicate his actions for this turn. He can resolve the dice in whatever order he likes, however all the dice of the same type must be resolved before resolving another type.
You must deliver a driver that demonstrates both the 1) Roll the dice and 2) Resolve the dice capabilities.

Part 5: Main game loop: Move

Provide a group of C++ classes that implement the move capabilities following the official game rules of the KING OF NEW YORK game.

- If there is no one in Manhattan, the player must move there. When he arrives in Manhattan, he must place his Monster on Lower Manhattan on the 2–4 space.
- If there is already 1 Monster in any zone of Manhattan, the player has two options: he can move to any borough that doesn't already have 2 Monsters in it (except Manhattan), or he can just stay in his borough.
- If he was already in Manhattan, he needs to advance to the 2–4 space in the next zone up in Manhattan.

Note:

- Once the player is in Upper Manhattan, he no longer move during this phase.
- If the player was already in Manhattan, he may not move to another borough (unless he has been damaged by another Monster's attack).

You must deliver a driver that demonstrates that 1) only a valid move is allowed, 2) the player can go with an optional or mandatory move

Part 6: Main game loop: Buy Cards

Provide a group of C++ classes that implements the cards buying capabilities following the official rules the KING OF NEW YORK game.
After resolving the dice and making a move, the player can buy one or more cards from those available.
The player also welcome to spend 2 Energy cubes in order to discard the three available cards and reveal three new ones.
The player can buy and/or discard cards in any order he likes, and as many times as he wants, as long as he has the Energy cubes to do so. Each time the player buy a card, immediately reveal a replacement for it from the deck.

You must deliver a driver that demonstrates that 1) the player can buy one or more cards.2) the player can spend 2 energy cubes. And 3) the player can buy and /or discard cards in any order and as many as he wants as long as he has energy cubes.

Assignment submission requirements and procedure

You are expected to submit a group of C++ files implementing a solution to each of the separate problems stated above (Part 1, 2, 3, 4, 5, and 6). 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.
You have to submit your assignment before midnight on the due date via Moodle. 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)