Concordia University

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

COMP 345 --- W2019
Assignment #2

Deadline: Mar. 25, 2019 by 06:00AM (**strict**)

Type: Team assignment (4 members max.)

Evaluation: 10% 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 four 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 *POWER GRID* game by allowing the player to:

1) Select a map (a map is separated into 6 areas, each with 7 cities) from a list of file(s) as stored in a directory.
2) Select the number of players in the game (2-6 players).
3) Each player choose which areas they will play in. It is recommended to have one area per player. Of course, the areas chosen must be adjacent. During the game, every player can play in all the chosen areas.

The code should use the map loader to load the selected and appropriate map file. It should load all the game pieces: the deck of the summary cards, power plant cards, the houses objects, resources tokens, the money(elektro), and the "setp 3" card. Also it should create all the players and assign to the player an overview card, the wooden houses of one color and 50 Elektro. You must deliver a driver that demonstrates that 1) the game valid board map can be loaded and its verified (i.e. it is a connected graph, regions and cities etc.), and invalid map(s) are rejected without the program crashing; 2) the right number of players is created, and the right game pieces created.); 3) and the player possession.

Part 2: Game play: main game loop Phase 1 and 2

Provide a group of C++ classes that implements the main game loop following the rules of the *POWER GRID* game. The main game loop has five phases and constitutes one game round. In each phase, all players take their actions in the order specified for the phase before the game continues with the next phase. In this part you are implementing phase 1 and 2:

1. Determine Player Order
2. Auction Power Plants.

You must deliver a driver that demonstrates that: 1) every player gets turns according to game player order. 2) Each player is given the opportunity to offer a power plant for sale at auction. The player can buy at most one power plant. The player has a `Pass()` method, this one is used when the player chooses not to start an auction, and has an `Auction ()` method, this one is used when the player chooses a power plant for auction. You must deliver a driver that demonstrates: 1) the player order phase; and 2) the possession of each player at the end phase 2.

Part 3: Game play: main game loop Phase 3 and 4

Provide a group of C++ classes that implements the main game loop following the rules of the *POWER GRID* game. During the main game loop that has five phases. In each phase, all players take their actions in the order specified for the phase before the game continues with the next phase. In this part you are implementing phase 3 and 4.

3. Buying Resources, where the players can buy resources for their power plants from the resource market. This phase is played in reverse player order.
4. Building, where the players start or add cities to their networks on the map.

You must deliver a driver that demonstrates that 1) the buying resources phase and 2) the building phase and 3) the possession of each player at the end of phase 4.

Part 4: Game play: main game loop Phase 5

Provide a group of C++ classes that implements the main game loop following the rules of the *POWER GRID* game. During the main game loop that has five phases. In each phase, all players take their actions in the order specified for the phase before the game continues with the next phase. In this part you are implementing phase 5.

5. Bureaucracy, where the players earn cash, re-supply the resource market, and remove a power plant from the power plant market, replacing it with a new one from the stack.

You must deliver a driver that demonstrates 1) the implementation of phase 5 of the game; and 2) each player possession at the end of this phase.

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, and 4). 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.

Important Note: For your demo set only one driver, and comment/uncomment the parts that you want to demo.

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 the due date using Moodle under *A#2_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)

.