Assignment 2

Due Sunday by 11:59pm **Points** 100 **Available** after Mar 30 at 12am

- This is NOT part of the team project. Do it on your own!
- Remember to run "git pull" to get the latest version of "dominion". **DO NOT make any changes to the "dominion" folder. (trunk/dominion)**
- If you have not checked in your directory to the class Git repository, follow the instructions mentioned in "Course Document" section to check in directory YOUR-ONID-ID/ to the repo.
- submit your work under class repository under your onid folder. Just like instructor's folder and file structure. (projects/christia/dominion)

CS 362: Assignment 2

Notes:

1) In this course, we test an implementation of Dominion card game. A directory named “dominion” in the svn repo contains source code for Dominion card game. The primary goal of this assignment is to familiarize you with Dominion and the source code.

3) Remember the class repository. **https://github.com/amchristi/cs362su2015** **(https://github.com/amchristi/cs362su2015)**

Tasks for this assignment (All tasks are mandatory)

Only for summer 2015 class: instead of committing .txt files, commit .c files. For example, instead of committing refactor.txt, commit refactor.c file.

1- (20 points) Create your own copy of the base dominion code, including everything required to compile the code (i.e. copy it from dominion directory in class repository to your own folder). Commit your folder in class repository. (Follow instructor's folder structure, projects/christia/dominion, your code should appear in projects/YourONID/dominion folder). **In case you have done this as part of week 1 course content, you don't need to repeat this.**

2- (20 points) Read the rules of Dominion, and understand the game sufficiently to be comfortable with testing an implementation of it! If you search online, you can find the official rules and multiple web sites allowing you to play the game for free, as well as forums discussing rules questions for most cards.

I used this website to learn dominion : **https://www.playdominion.com/ Dominion/gameClient.html** **(https://www.playdominion.com/Dominion/gameClient.html)**

Your first job is to become a “subject expert” in Dominion, since you will be testing an implementation of it. Note that the primary source of information about the Dominion implementation itself is the dominion.c and dominion.h files provided in the class repository. You have to combine the knowledge you acquire by playing the game with dominion code to fully understand the specification of the system. This is a typical testing experience, where you are not given a complete specification, but must discover one for yourself.

I have provided two items, "dominion base architecture" and "dominion documentation to start with". It provides you an explanation on how to run dominion code.

Submit a file called **documentation.txt** that contains documentation of **smithy, adventurer** cards. Documentation should contain your understanding of smithy and adventurer cards (code). It should also contain documentation of your understanding of **discardCard**() and **updateCoins**() method. Keep your documentation short, though there is no upper limit. Documentation.txt file should contain at least 100 words.

3- (60 points) Pick five cards implemented in dominion.c. Choose 3 cards of your choice and smithy and adventurer cards are mandatory. Refactor the code so that these cards are implemented in their own functions, rather than as part of the switch statement in cardEffect. You should call the functions for these cards in the appropriate place in cardEffect. Check in your changes, with appropriate git commit messages. Document your changes in a text file in your dominion source directory, called “**refactor.txt**.” Your implementation of at least 4 of these 5 cards should be incorrect in some way, i.e., you should introduce subtle bugs that are hard to catch in your changes. Introducing bugs in smithy and adventurer is mandatory. Write information of your bugs also in refactor.txt. Later in this class, other students will test your code, so try to keep your bugs not superficial. Refactored program should compile without any error.

IMPORTANT: Only commit to your <onid-id> directory. DO NOT modify other people's directories or the dominion base directory.

All the updated code, documentation.txt and refactor.txt file should appear under your onid/dominion directory.

More Notes:

• If you have not checked in your directory to the class Git repo. Follow the instructions in "Course Documents" section.