

Assignment 2

1) Create a new branch from Master (similar to Assignment-1) for Assignment-2.

– Done. Branch is “31989226-assignment-2”.

2) Read the rules and the source code of Dominion, and understand the game sufficiently to be comfortable with testing an implementation of it!. 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. The specification you use will have to combine this information with knowledge about how the game works, discovered by investigation. This is a typical testing experience, where you are not given a complete specification, but must discover one for yourself.

Done. While no work is shown, I opened up *dominion.c* and *dominion.h* to review the functions for implementation of the game, specially the cards, prior to Q3.

For 3-5, see the next page. I have documented my work / responses respectively, ie 3 = Refactor and 4 = Bugs + Screenshot of compiled and run code, 5 = as is.

3) Pick **5 cards** implemented in *dominion.c* (see the *cardEffect* function). Choose **3 cards** of your choice plus the **smithy** and **adventurer** cards which 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*.

4) Introduce some bug(s) in **4 cards** out of these **5 cards**, preferably “subtle” ones that might easily escape a decent test suite. By bugs I mean something that does not behave correctly – it may crash, or it may cause incorrect Dominion behavior. Introducing bugs in **smithy** and **adventurer** is mandatory. **ALL CODE SHOULD BE COMPILED and RUN.**

5) Document your changes of the five cards in the **Assignment-2.pdf** file, under a section called “**Refactor**”, discussing the process of extracting the functions (**35 points**). In addition, write information of your bugs in a section called “**Bugs**” (**35 points**).

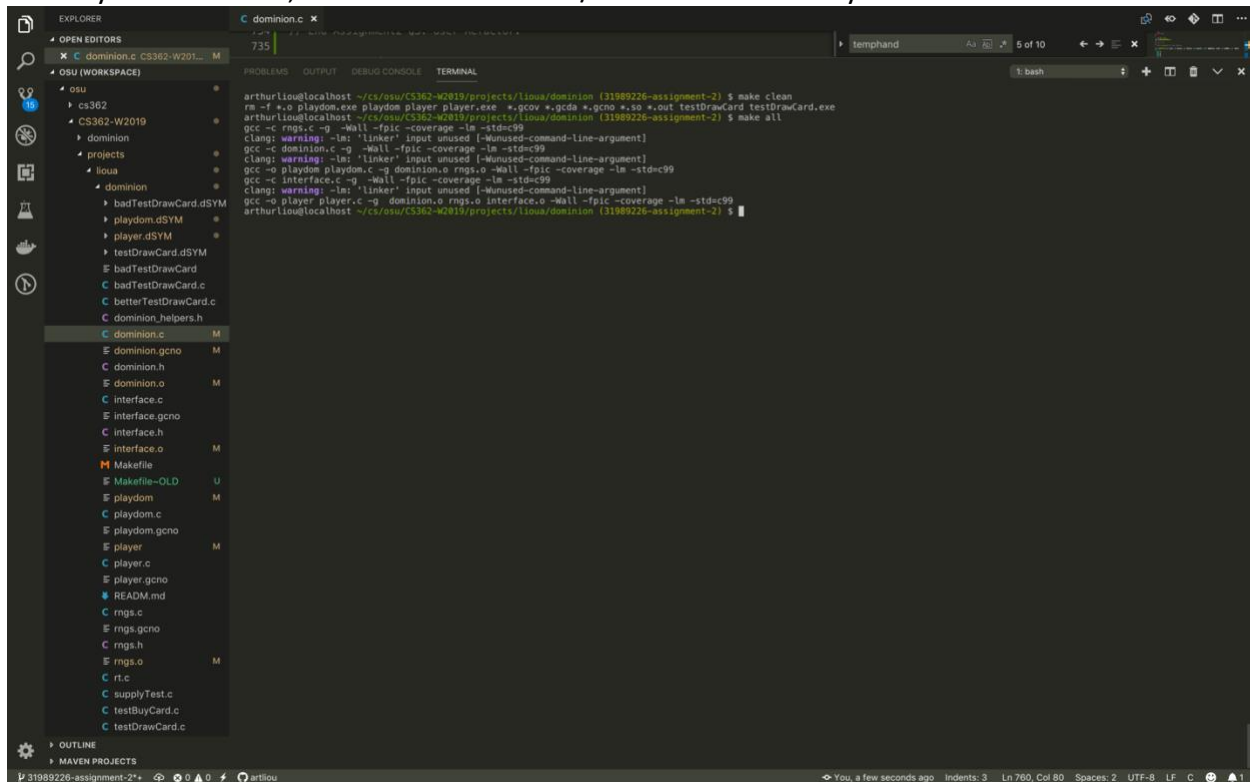
Refactor (ie. Part 3)

Mandatory Cards: Smithy (831), Adventurer (669)

“My Choice” Cards: Village (842), Remodel (805), Gardens (767)

Note: The numbers in parentheses are the line in the original dominion.c file the card’s switch case is found on.

For my refactor, I picked these five cards: Smithy, Adventurer, Village, Remodel, and Garden. For each of the card’s switch case within the cardEffect function on line 736 (revised dominion.c), I removed the code / functionality from the switch case and place them in their own function(s) starting around ~Line 646-734, ensured that I passed in the correct parameters and any initializations, and ran a make clean/make all to show my code runs.



Side note: I’m not sure if this was intention by staff who created the source code, but the indentation and spacing for are all off / very confusing, which made tracking the start / end of each case/function block very annoying.

Bugs

Format: Card – Bug Description (Line #). Bold is mandatory

1. **Smithy** – Draw 4 instead of 3 (655)
2. **Adventurer** – Excluded Gold from Treasure Drawn (674-675)
3. Villager – Added 3 Actions instead of 2 (695)
4. Garden – Allowed a Player to Draw a Card (731-733, 861-862)

I’ve attached two screenshots on the next page of my code being compiled (make clean, make all) and run (./playdom 30).

Left - Local
Right - Flip1