Testing Report

Dominion, as I know, is very popular card game, but it is my first time to hear that. Therefore, during writing testing programming source files for debugging dominion, I had met quite a few difficulties. Basically, I did not understand all regulars to play this game very well. Most time, I asked some details about how to play this game from my friends. In general, all source files about Dominion is hard to test because all files have more than thousands lines so that it is hard for me to understand purpose of author. In addition, all coverages of relative code are very regularly low since they are not over 20% - 30%.

The first assignment was the easiest one. We just needed to pick up five Dominion card’s achievements and put them into five individual functions. Although it was easy to do, code lines of each card in switch() were very long. So, it was not convenient to read and understand. In addition, it seemed that there are some card’s achievements had similar structure. That is the reason why that assignment let us pick up five card’s achievements into five individual functions.

The second assignment was harder than the first one. The main reason was that it was very first time to touch and use unit test to debug each uniform function. Seemingly, we just needed to write several lines of code to test, but I thought that was hard to write. In fact, when I imitated sample code to write code for four unit tests and card tests, it did not show any meaningful testing results. Seriously, if could not use function assert() to limit testing conditions, testing code would not execute due to result of assert(). Usually, entire testing process would be terminated immediately. Moreover, to generate coverage for source code is novel for me. Actually, I did not know C programming compiler has this kind of function. As a result, I had struggle to use command gcov to measure code. But, I conquered to use gcov to measure coverage by search relative articles in the Internet for help. Therefore, when I tested those unit tests, it sometimes shown 100% as coverage or sometimes it would not execute result from assert().

The third assignment was random testing for two cards’ achievements. Just like I have mentioned above, all supported source code about Dominion had too many bugs so that it was hard to guarantee when this game could execute. Random testing was used in this situation and it was novel for me as well. After I wrote all of my testing code, I tried to run them and the coverage outputting files were not generated always. In fact, during running those testing code, terminal always shown segmentation fault after several rounds of testing. Even though it could run to final status, the coverage shown in generated file was very low, which was usually 30% - 40%.

The fourth assignment, I think, was the most difficult assignment. It requested us to test entire Dominion game. Initially, I did not know how to test the entire game because I always thought that this game could not run since there were too many bugs. Also, that assignment requested to generate several random numbers for player’s number and random cards. According to sample source files, playdom.c and playdom2.c, I tried to stimulate style to write my testing code for assignment 4. But I could not execute and output correct results because I failed to understand meaning of returned values of some operating cards functions, such as shuffle(), initialGame(), and so on. Those functions returned Boolean value to decide whether those functions were executed successfully. However, I thought those functions returned enumeration of cards. Finally, I asked my friends for help and then solved how to use those functions to initial game and play game, and print out result of playing game.