Isfandyar Rabbani, 932960451
CS 362
Assignment 5

Bug-Report

Bug # 1 – Smithy adds 4 cards to the hand rather than 3.

- Found by cardtest1.c, randomtestcard1.c, and randomtestcard2.c.

Bug # 2 – Council Room adds 5 cards to the hand rather than 4.

- Found by cardtest4.c, randomtestcard1.c, and randomtestcard2.c.

Bug # 3 – Village adds 3 to the action count rather than 2.

- Found by cardtest2.c, randomtestcard2.c, and randomtestcard2.c.

Test Report

I am completing my final project on my own, so rather than a teammate's code, I tested another student's code that I found on GitHub. It is code from their version of assignment 2, so it has "subtle" bugs purposefully introduced into it. For this reason, most of my card tests and random tests were able to detect three of the bugs introduced. None of the unit tests found any bugs, probably because there were no bugs introduced into the system except for bugs in the individual cards.

The one concept that I will take away from this testing experience is the importance of maximizing the modularity of code for any given system. We were able to complete this assignment because the rubric of our tests can be applied to anybody's code. This demonstrates you can use your requirements to tailor any code to be useful to you, all with meticulous and thorough testing. Since these bugs are purposefully introduced into the system, I will refrain from making any sweeping generalizations about the reliability of their code.

Debugging

All of my tests are set up with strict assert statements, so any debugging was up to my own meticulous sifting through the code in the Visual Studio debugger. Determining what could be tweaked in a card's function so that it caused errant behavior was the first step in debugging a certain card. This was achieved by stepping through the test programs in VS one line of execution at a time, and then analyzing each variable's role and how the flow of execution played out.

Another debugging technique that helped me narrow down the source of bugs was to introduce checkpoint print statements every time a variable is to be manipulated once my tests asserted that something was indeed wrong with the program. Right before a statement where a function is called, I would print to console the value of the variable and what the expected result was. Then, following the function call, I would print to console again the actual result. This made it crystal clear where the culprits lay.

My debugging procedure revealed the logic as to how the bugs were introduced in this code. Implementing the bugs was achieved by increasing critical values by 1. These values controlled while and for loops and was also an integer that was being added.