








i. Section Tool:

1.

dominionMaven		87.1 %	1,975	293	2,268
src/main/java		87.1 %	1,975	293	2,268
dominion		87.1 %	1,975	293	2,268
Card.java		83.7 %	999	195	1,194
dominionBoard.java		89.5 %	769	90	859
dominion.java		0.0 %	0	8	8
Player.java		100.0 %	207	0	207

The generated tests covered my code fairly well. My unit tests covered it by at least 90% so it's not as good as my tests.

2.

Evosuite didn't find any bugs I already knew about.

3.

Just getting it to run was a nightmare and took most of the time for this assignment. When I finally got it to generate tests some of them crashed before it even got to my code.

4.

In the time I spent getting this to work I could have written my own better tests. It found ways to break my code but only because it setup the variables in normally impossible ways.

5.








I looked at all the tools listed on the lecture slides. Evosuite was the only one I really tried using.

6.

I picked this tool because it was the only one compatible with eclipse. The tool itself took ages to get working. Once it was working it did generate a decent amount of tests that got ok code coverage. The tests themselves were mostly useless. Some of the tests failed before touching my code. Some of the tests were just flat out wrong. Some of the tests weren't even tests. The only time I would

think to use this tool again would be for security testing; it tries to break your code in the way that someone trying to break your code would do.

ii. Section Random Tester:

dominionMaven		94.0 %	2,131	137	2,268
src/main/java		94.0 %	2,131	137	2,268
dominion		94.0 %	2,131	137	2,268
dominionBoard.java		92.9 %	798	61	859
Card.java		95.0 %	1,134	60	1,194
dominion.java		0.0 %	0	8	8
Player.java		96.1 %	199	8	207

My dominion was already setup to play automatically. The players randomly pick from a list of all possible options. For example, during the action phase the players randomly choose from all action cards in their hand. Originally my dominion was only designed for 2 players so I had to make many changes to my code to allow for 3 or 4 players. I then just ran a new dominion game a few times with a random number of players (between 2 and 4). The code coverage was better than the Evosuite but the tests aren't nearly as robust.

iii. Section Bugs:

Evosuite found a bug that there was no limit to the number of players the game would accept. I discovered this because the tests were trying to run with thousands of players and crashing due to memory issues. My code worked fine if not for the memory issue but I fixed it with simple input validation.