Mikkel Hansen

CS 362

Test Report

I tested galec’s code with my testdominion.c file. Using several tests, I was able to get a coverage of 79.9% in the dominion.c file. The adventurer and feast cards were not run at all, which caused the coverage to be lowered. Some specific lines in certain cards were not run either, contributing to lowered coverage. However, most cards were run in some manner, so I can say that other than the feast and adventurer card there are no program halting bugs. I used a whole-game tester so I could see the game state after each turn. Right off the bat, I saw that the feast card has a bug. This bug causes the code to go into an infinite loop when the feast card is played. The reason for this bug is because there is a while loop in the feast card that once entered, you cannot leave because the card does not give the user any opportunity to give new input so the loop will not occur. This is a major bug because it makes it so that the game is not playable without removing or bypassing the feast card. Another major bug that I found is a bug within the adventurer card. This card has several lines where it will seg fault, causing the whole program to stop. From my understanding, it has something to do with the variable “z” that is being used as an index to store cards. This is a major bug because it makes the game unplayable unless the cad is removed or bypassed. A less significant bug that is present in the code that I saw by glancing at the coverage is in the sea hag card. This card messes with the deckCount value of all players except the current player which causes cards to be added or removed from the game because the deckCount will be incorrectly incremented or decremented, adding or removing cards respectively. If the adventurer card and feast card are removed, we can get the game to run to completion each game. However, we do not know about the card logic that is implemented. So this means that we need to look at the state during and after each game in order to see if the card that was played did the correct action. We would also need to check the supplyCount before and after a card is bought. Overall, the reliability of galec’s dominion code is poor because it does not run to completion if feast or adventurer are played. Once these cards are fixed, then the reliability of the code increases dramatically because the code was run to completion each game in my tests. Fixing these cards would then allow more bug fixing by running the code more reliably and checking for logic bugs in the output files. One could also implement other testing strategies on the code such as tarantula. In terms of the status of the code, I would say that it is currently unusable, but that once the feast and ambassador cards are fixed then the code is very usable, and only minor logic errors need to be fixed.

I tested kaiserh's dominion.c file with my testdominion.c file. I used a whole game tester so that I could get information about the state after each players turn. This way I would be able to detect small bugs produced by playing a card. Running several tests, I got a coverage of 82.04% in the dominion.c file. This is slightly better than the last test I did on someone's dominion.c file, but it is still lacking in some areas. I was only able to get the code to run by disabling the feast card. The coverage results tell us that even though some of the cards were disabled, there is still code that is not being run. After looking through the gcov file, most of these un-run lines were in functinos that my testdominion.c file implements itself. However, there are some lines of code in certain cards that are not run, which is not a good sign. These lines could potentially get covered if many more tests were run. The largest bug in the code if the infinite loop that occurs if the feast card is played. The reason for the infinite loop is that if the feast card is played, the code goes into a while loop that is never exited, since the code does not ask for any new input and never breaks out of the loop. This is a major bug because it makes the game unplayable when feast is a usable kingdom card unless the code is removed or bypassed. Using the gameResults.out file that is created when the game is played, I was able to filed a couple other bugs as well. There is a similar bug in both the mine and remodel cards that has to do with a ">". The bug is that the code calculates the cost of the cards to be swapped, but does so incorrectly due to the ">" sign. If this ">" was a "<" instead, then the code would correctly calculate the cost and not incorrectly return -1 when it should really swap the cards. Another bug is in the sea hag card. The sea hag card will incorrectly decrement the deckCount of everyone except the current player, which causes cards to be created or destroyed in the deck because the code relies on the fact that the deckCount is correct in order to get the index of the top card. If the index is too small as in the case of the sea hag card, then there will be too few cards in the deck, meaning that the sea hag card "destroyed" some cards. Overall the reliability of kaiserh's code is medium because there is an infinite loop bug, but it is the only bug that will halt correct functionality of the code. Since there is only one such bug, the code runs successfully more often. The rest of the bugs are minor logic bugs, and although they affect the state of the game, they do not halt or crash the game, so in terms of reliability they do not matter much. In terms of the status of the code, I would say that the code is semi-usable because there is only one game-stopping bug, and once the bug is fixed then there are only minor logic bugs to fix.