James Carlin
Aditya Gune
Brian Tiegs

Final Project Part A

Explain testIsValid Function of UrlValidator test code.

The function works by building a url, running the url through the UrlValidator.isValid() method and asserting if the method produced the correct response.

The testIsValid() function completes each url by looping through the resultPair arrays: testUrlScheme, testUrlAuthority, testUrlPort, testUrlPathOptions, and testUrlQuery. Each part of the resultPair arrays is appended to a string and the expected value is calculated. Once the url is built, it is used to run the UrlValidator.isValid() method. The UrlValidator.isValid() method returns true if the url is valid and false if the url is invalid. An assertEquals statement is used to compare the expected value with the returned value. If the values being compared in the assert statement are not equal then the test fails. The indexes for the ResultPair arrays are incremented and the test repeats. This continues until all possible combinations have been iterated through.

Explain how it is building all the urls.

A url is composed of multiple parts: scheme, authority, port, path and query. In order for the url to be correct, all parts must be valid. The function builds a url from multiple ResultPair arrays where each array represents each part of the url. There are 5 ResultPair arrays: testUrlScheme, testUrlAuthority, testUrlPort, testUrlPathOptions and testUrlQuery. Each element in the ResultPair array contains a url part along with a Boolean stating if the url part is correct.

The function builds the array by selecting a url part from each array in the correct order and appending it to a string. The Boolean value associated with each part is compared. If a false statement appears at any url part, the url equates to false. This value is the expected outcome of the UrlValidator when the url is used.

Give how many total number of urls it is testing.
The testIsValid() function tests 31920 urls.

Give an example of valid url being tested:
`http://www.google.com:80/test1?action=view`

Give an example of invalid url being tested:
`://255.com/../?action=edit&mode=up`

Do you think that a real world test (URL Validator's testIsValid() test in this case) is very different than the unit tests and card tests that we wrote (in terms of concepts & complexity)?

Conceptually, both the student and the real world test are similar. The concepts we've learned in class were applied to the testisValid() function. In our tests and the real world tests, input domain partitioning was used to construct meaningful tests. In addition, both use an oracle to determine if the output is correct based on the specification.

As far as complexity goes, our unit tests seemed much more complex. The UrlValidator.isValid() method is simple and requires only a url string as an argument and returns true if the url is valid or false if the url is invalid. The dominion.c functions require many more arguments including the complex struct gamestate and each function could has the ability to alter the gamestate. For this reason, the input domain for the dominion.c functions is much bigger. This also presents a greater challenge when checking for the correctness of specification. Since so many variables can be altered, the oracle would also need to be more complex to check all of the variables for changes.