Summary

The overall Peer Review process was extremely useful for debugging and recognizing improvements we could make in our game. There were tests on our code that we didn't think of implementing but that other groups did and found issues in our program with. As we had pre-existing biases that our own code would be correct and function ideally, testing on ourselves proved to work in some cases, but it was definitely not as thorough as the Peer Reviews we received. For these reasons, trying to break someone else's code was definitely easier than breaking our own code.

Writing our test code and finding inputs to test the other groups' code allowed us to think about specific ways that we might break their code. We were able to think about the consequences of performing tests with those inputs and translate that to our own code. For example, one of the tests we used was creating a board with a large size. At a certain point, the board will no longer be able to make cube configurations to fulfill the large board size. By testing cases like these, we were able to better understand how our program code should go about handling these cases and also encouraged us to have greater knowledge of how individual components in our program should interact.

While testing the other solutions, we were able to more clearly define which cases would be considered edge cases. When some inputs would have unintended behavior on other group's programs, we were able to recognize why those inputs in particular cause that behavior and take educated guesses for why they might have gone unhandled within someone's program. For instance, while we were testing one group's board, we noticed that the board environment highlighted two instances of the same character when it should have been only one. We knew that something must have been working improperly with the way their program searches for

words. Because of this, we went back to our own code and tested if our search and add word functions behave like that as well.

From the Peer Reviews we received, we were able to see which tests that other groups thought of that caused our program to fail that we might have missed. For example, one group noticed that there was an inconsistency between the number of dictionary searched words and the number of board searched words, as one returned 195 words while the other returned 198 words. While we might have caught this issue later on, we did not think to test this in our initial testing. The Peer Reviews also gave us another perspective of how the game should operate. Even if a lot of the feedback we received were design decisions, having feedback from people other than us allowed us to improve the user experience with the UI.

Evaluations

Ranking of reviews (from best to worst):

1. Group #14
2. Group #21
3. Group #6
4. Group #29

Group #14 and Group #21 were the most useful because the issues that they pointed out were fairly significant errors on our part. They also talked about edge cases that we were not aware of. Group #6 and Group #29 pointed out issues in our code as well, but the issues were smaller and focused on design decisions. Group #14 and Group #21 also directly addressed our program and made their explanation targeted towards what we had written, while Group #6 and Group #29

felt like a significant portion of their description could apply to different programs as well. The goals and results were clear to us for all the reviews, but some of them were more specific and thus helped us debug our code faster. One of the things done best for all of the groups was that each of their testing methodologies was extremely thorough. There wasn't a group that made it feel like our code wasn't thoroughly played enough, and the reviews definitely helped us identify and fix bugs, which we outlined below.

Revised Solution

Based off both the feedback from the peer review, we looked at the tests we failed or just overall suggestions they had and made the following changes:

- Distinguish between guessed and invalid words
- Make play again or exit prompt at end repeat until a valid argument is selected
- Account for inputting null to isPrefix or contains for GameDictionary (used to give NullPointerException)
- When the dictionary has not yet been loaded, iterator.hasNext() was returning true so we fixed that edge cases
- Handle empty lines in cubes and words.txt
- Printing out all possible words with the correct statement (used to say the wrong thing English-wise)
- Updated board search, found small bug that made it miss a few words that dictionary search found
- Additional input verification and handling issues when newGame or loadDictionary isn't called but other methods are

- Updated lastAddedWord to return points sequentially in the order of the word instead of in random order

Additionally, based off what we saw worked well in the UI of other groups' projects when we were testing them, we made these changes:

- Query cubes and words.txt at beginning
- Display computer score at end
- Case insensitive inputs for everything
- Display the player winning at each turn