Functionality: Our keyboard input functionality is decently robust. It only allows for appropriate inputs and blocks those that do not work. However, with larger sized crosswords with dimensions greater than 4, the program takes too long to be effective. In such a case the user should abort the program. We also have issues with silly symmetric crosswords that we could eliminate with more time.

Design:

Creativity: Our idea for this assignment was not the most creative in the world. This concept has been explored ad infinitum with other assignments. The unique concept in our implementation was the usage of the Depth for Search algorithm, which is more of a design feature than a creative one

Sophistication: Our design is quite sophisticated. By looking at the program specification document, you can see how it first populate a grid with a completed crossword using the word bank specified by the user. It then prints out that grid in a jpanel, resulting in a completed crossword. We also used one of the algorithms we learned towards the end of this semester in Depth First Search to create correct crosswords.

Broadness: We checked off all the items on the list except for 3. For 1, we used the regex library for a lot of the functionality of our code. For 2, we defined DepthFirstSearch as a subclass of GeneratedGrid class. Our user-defined data structure is the node system we set up for the DepthForSearch algorithm, while the ArrayList we used to store the Cell classes in GeneratedGrid was a built-in data structure in addition to the stack in DFS. We also used file input for the wordbank .txt files. Randomization was also key as it was how we generated our crosswords. Finally, we needed to use generics with our ArrayLists, as ArrayList is a generic class.

Code Quality: Our code is clear and well-commented. Our variable names are easy to follow and the flow of the program is easy to understand with the readme file.