CSC221 — Lab #3
March, 2020

Boggle™ is a board game, for more information about the details of the game please see `https://en.wikipedia.org/wiki/Boggle`. The game will be demonstrated in class. Essentially, the game is about finding as many English words in a matrix of letters as you can. Of course, there is nothing special about English words, but our solution will only focus on English words. A score is computed based on the words found. Of course, the goal of the game is to score as high as possible.

This lab introduces a *retrieval tree*, a **trie** (pronounced "T-R-Y"). You must modify provided the Java code used in a provided application that plays Boggle. Probably most of your time will be invested in studying and understanding the supplied code. You **must** get the template assignment from gitHub Classroom, using the *clone* operation. If you have had difficulty retrieving past templates, please get my assistance with when you download the lab template.

The provided code contains a complete solution to Boggle. You want to modify this code to use a **trie** to significantly improve the runtime complexity of the program. A **trie** is a mutiway search tree, where the data is stored on the paths starting at the root.

As the search for words progresses it will be necessary to check to see if a string found so far is either an English word, is a prefix of an English word, or is not the prefix of an English word. We require a set of valid English words to help in this determination. The provided code includes a file (*twl06.txt*) of the officially accepted Scrabble™ words. In the provided code, these words are read into a *Set*, represented by a *TreeSet*. You will need to carefully and delicately modify the code to use a Set represented as a *Trie*. In the provided code, a *Trie* class is provided, which you **must** modify. **You may not substitute another Trie class for the one provided.**

Also included in the provided code is the class *Grid*.

Words are found in Boggle starting at any cell and moving to any of the eight adjacent legal positions. Essentially we will be using a brute force technique with *backtracking*.

All changes that you make to the supplied code must be well commented. You should claim credit for all changes as well.

This is a **pledged work** assignment. The work that you turn in for grading must be your own.

This assignment must be turned in through gitHub classroom no later than 5pm, on March 24, 2020.