**Contagion – Option 2**

Fahed Alrafati and Mary B. Makarious

**Primary Goal**

The goal of this project is to use quaternary trees to implement a search method that can find a given 30 base pair genetic sequence within a given genome. This search method must also account for a 10% error rate, meaning that at most three of the 30 base pairs can be incorrect. This can be implemented using a recursive method that calls the search three times for every error, offering every alternative search path for that error.

**Data Structures**

The primary data structure that was used was a quaternary tree. A tree is one of the best methods to implement a search method upon, since its search time in O(log n) where n is the number of data points in the tree, making it very efficient and ideal for our project. Another data structure that was used was a vector. The vector holds the various words that will be searched for in the tree as they are extracted from the file provided. Another vector also holds all the possible words that exist in the genome, so the tree can then be created from them. Later on, the vector will be printed out into the results file.

**Steps for the Program**

To run this code, the correct files must be entered. One of these files must contain the genome, while the other file contains the 30 base pair reads that will be searched for. The program will then go through the reads to identify which ones are found and which ones are not. The information will be displayed to the console as well as written out to a file.

The tree will be traversed according to read that is being searched for. When there is a node missing in the tree (when an error is found), the search is terminated and repeats again from that same error, assuming that the incorrect base could have been any of the other three bases. For example, the program will search for “aaactg” in the tree. It finds that it can trace this sequence through the tree until the fourth base pair, “c”. So it stops the search, and repeats it three more times assuming that the “c” could have been “a”, “t”, or “g”. Thus, the program searches for all of “atg”, “ttg”, and “gtg” from the same point in the tree.

The issue with this method is that if the error exists early on in the searched read, within the first 6, the program will not assume that there is an error there. This is because the tree is denser towards the root and it would be impossible to find an error within the first about five base pairs. The program will thus find an error later on and report a false negative.

**IDE and OS**

OS: OS X El Capitan

IDE: XCode Version 7.3

**Division of Labor**

Fahed Alrafati: Wrote code, implemented the quaternary tree as well as designed the recursion class that allowed for 10% error rate. Managed debugging and reviewed final project overall.

Mary B. Makarious: Aided with debugging, cleaning up code redundancies, commenting and preparing documentation.