My project is concerned with finding the degree distribution of this dataset of Enron employees' email correspondences. It is an undirected graph of 36692 nodes and 367662 edges, with the nodes corresponding to different email addresses and each edge as at least one email shared between two unique addresses. My goal for this project is to get a sense of whether the employees of this large company tended to work in larger teams or one-on-one with other people, as shown by their email habits.

I first started out with setting up my Github so that I could push to my repository. After initializing everything, adding all the necessary files, and using the correct terminal commands, I began the real work of coding.

Within my file, I first started with creating a file reader function to read the email-Enron.txt file into a HashMap containing the nodes as its keys and the nodes each node is connected to as its entries (as shown as a Vector of nodes). I read the file into one large string, split it into separate lines, and trimmed off the lines that weren't necessary (such as the empty line at the bottom and the few lines of text explaining the dataset at the top). Then, once I had all of the lines that were relevant to me in one large vector of vectors of numbers, I began to finally assign each individual node into the HashMap along with a vector of all of the nodes it's connected to as its entry. This process was made easier because I knew that all of the nodes were some value between 0 and 36692 since this is what the file had told me at the top of the document. Therefore, I was able to utilize ranges to make this process simpler. At the end of this function, I returned the HashMap with the nodes as its keys and a vector of connecting nodes as its entries.

After that, I began to do calculations on this HashMap. I created a function to get the total number of degrees of each node from the library I had created in the prior function. This function, degreetotal(), returns a HashMap with the nodes as its keys and the nodes's degrees (or total number of edges) as its entry. Additionally, the function also returned a vector full of all of the degree levels of each node of the whole dataset. This will be useful in my next function.

My next function is where the rest of my computations lie. Within this function, I first created a new HashMap that switches the places of the key:values of my previous HashMap (which holds node:degree). Instead, I now take the degree level of my nodes and tally up how many other nodes have that same degree level. I placed these values in a new HashMap (so this HashMap holds degree:number of nodes with that degree). After that, I take my vector of degree levels from before, sort it, and remove duplicate entries. I then create another HashMap that holds each degree and its frequency within the dataset (degree:frequency) as well as a vector that holds all of the frequencies. Depending on a third input given to the whole function, the function will then either print each line of the degree distribution HashMap I have created in the order of smallest to largest degree, or it will print the top most frequent degrees of the distribution. The input will specify how many of the top distributions to print out (for example, if the input was 17, it would print the top 17 most frequent degree levels including ties).

After outlining all of my functions, I then created a way to enter user input into my program. If the user types “all” (capital or lowercase) into the terminal, the program will print out the entire degree distribution of the dataset. If it types an integer between 0 and 74 (inclusive) as instructed, it will get that number of the top distributions of the dataset. If what is typed is not an integer, the program will give an error that says “Not an integer number” and end the program. If what is inputted is a string that isn’t “all” or an integer number not in range, it will ask the user to try again with inputting the values specified. I achieved this by using a while loop with nested if statements that call my functions.

To end, I split my functions into modules and created tests. The first test ensures that all of the frequencies that are held within my frequencies vector are under 1, since no degree has a 100% probability. My code passed this test. My next test was to ensure that the number of degrees I had found (as shown in the length of my degrees vector) was equal to the number of nodes that were in my dataset. This is to ensure that I haven’t missed any nodes; this test also passed. My third test was to ensure that the number of degrees for each node was larger than 1, since if the node was in my dataset, it was connected to another node. This is to ensure that my degrees are accurate, which was true for my code. My fourth test ensured that when I added up all of the degrees I had in my vector containing them, they equalled 367662, which is the number of edges that my dataset had to begin with according to the document I was given. This test passed. My last test made sure that when I added up all of my frequencies, the sum was close to 1. Due to computer error and very small decimals that are prevalent in my dataset, I made sure that this value was close to 1 and not 1 exactly since it’s practically impossible that my sum would be 1 exactly. The sum was higher than .99 and lower than 1.1, which is close enough to 1 that my test passed.

To run my code, you either type “all” into the terminal or an integer between 1-74. “All” will print out each line of my degree distribution by degree level. The positive integer will print out that number of the top frequencies and their corresponding degrees into the terminal. Other inputs may raise up one of the errors/exceptions I’ve created or prompt you to try again.

From this process, I’ve learned that Enron prefers for their employees to work in very small, often one-to-one teams, as shown by the larger frequencies of degrees when degrees are lower. Moving forward, this data is interesting because it can help improve workplace habits and team dynamics of other, similar companies.