First, I needed to find some examples and tutorials of JavaMail API. There were a few sources I found on Oracle, YouTube, as well as TutorialsPoint.com. However, I was not used to using .jar files and including them in my classpath. It took me a while to figure out how to run it and how the syntax differs between linux and windows. For example, compiling the program with windows would have a format such as "javac -cp mail.jar;activation.jar;   myProgram.java", where semicolons are used rather than colons. Furthermore, it took me some time to figure how to run the compiled program with the classpath (java -cp mail.jar;activation.jar; myProgram).

Once I could get examples I found on TutorialsPoint.com running, I could find components in the program to run based off the assignment. Originally, I was going to follow Oracle's guide on JavaMail, however, they ran it off of NetBeans. Since, I preferred to run the program off the command line, I used TutorialsPoint: https://www.tutorialspoint.com/javamail_api/index.htm. This was extremely helpful as it outlined many aspects of the JavaMail API.

Like most programs, I did come across a few challenges on the way of developing my program. Some were a bit trivial, but important nevertheless. To be honest, I wasn't really sure what the difference between a CC and BCC recipient was, so I just had to look it up. It was quite convenient that the JavaMail API could easily handle that. For question 2, I had to adjust my program so that it could accommodate attachments. Reading some of the documentation, I decided to send the text and attachment using "multipart" and then setting the content of the message using multipart. Had I done it without, the attachment would have overwritten the text.

I could run the programs successfully first for question 1 and 2, when the emails were sent to the right recipients, with the right CC or BCC according to the .txt file. Attaching an image would not overwrite the main body of the message. For question 3, I could retrieve the messages trying out various emails. Again, I was quite fortunate that the JavaMail API is well developed and well documented. As a result, I did not come across major issues when debugging and testing my program.

The overall design for my first program would be to just make one program for questions 1&2. Each recipient to, cc, bcc, would have an array, so that the list in the .txt file could be read and set in their respective arrays. In order to read the .txt file, I used Scanner to retrieve each line in the file and see if it begins with the heading followed by a colon (eg. Subject:). If it did, I would parse the string and obtain the second part as the String to set the variable. With the case of the recipients, I parsed that string by seperating the commas, and taking the length of the parsed string array and setting the length of the recipient array. Then, I could set each recipient accordingly. For the body, I switched a boolean to true once "Body:" was read. This would allows my stringbuffer to append each line thereafter as the body of the text. All the parameters would be passed into my method to send the email. Each recipient was defined accordingly (TO, CC, BCC). Each part of the email (attachment and body) was defined by multipart and passed to the message before it was finally sent via transport.

The overall design for my second program was to have the user enter 3 or potentially 4 arguments for the server, email, password, and email index to look up. I used mailstoretype as pop3, and that did limit the types of servers I could use. The method found the email INBOX and took out the messages as an array. Iterating through each array, it printed the subject and sender. If the 4th argument was entered, it would iterate until it found the email of interest and printed its contents.