14.07 Assignment Instructions

Instructions: Write a program to perform a frequency analysis of the letters in a message and use the results to decipher a secret message.

Part 1: Frequency Analysis Lite

1. Be sure you have downloaded and read the 14.07 Virtual Lecture Notes (Part 2) to the Mod14 Documents folder.
2. Create a new project called 14.07 Frequency Analysis in the Mod14 Assignment Folder.
3. Create a FrequencyAnalysis and a FrequencyAnalysisTester class in the newly created project folder.
4. You will need a plaintext document of approximately 500 words or more for this assignment. The *plaintext.txt* file downloaded previously is a good example, but you may choose to use a paper or essay you have written for another course, or you may copy and paste something from a web site. The longer the better so the frequency distribution of the letters will be more accurate.
5. Remember to verify your program first, using a subset of the document that is small enough for you to manually count the letters. Simply copy and paste the first 15 – 20 words from your text into Notepad and save the subset of the document as *subset.txt*. Count how many times each letter occurs in the *subset.txt* file and calculate the percentages with a calculator. You will use this information to verify that your program works correctly.
6. Read the *subset.txt file* and determine the frequency of the letters in this simple plaintext document.
7. Print the results neatly to the screen (e.g., control horizontal spacing, alignment, and decimal places). Select the Unlimited Buffering option on the Terminal Window menu, so all of the information will appear.
8. Write the results back to a separate text file called *subsetfreq.txt* that shows the letter, the number of times the letter was found, and the percent frequency of the letter. Each row in the file should represent a different letter.
9. Your program should be able to read a text file of any size. If your program works for the *subset.txt* file, you can be reasonably sure that it works for large files.

Part 2: Frequency Analysis of Plaintext

1. Once your program works with the *subset.txt* file, it should work for a text file of any size. Substitute the *plaintext.txt* file or a text file of your choice, for the *subset.txt*.
2. Be sure to also change the name of the output file from *subsetfreq.txt* to *plaintextfreq.txt* so the new data will be recorded without overwriting the old *subsetfreq.txt* file.
3. When you run the program, the frequency analysis of the letters in the *plaintext.txt* file should be printed neatly to the screen. Copy and paste the results into a word processor and print a hardcopy of the results. Be sure to label these results as the Plaintext Frequency Analysis to avoid confusion.

Part 3: Frequency Analysis of the Ciphertext

1. Download the encrypted *ciphertext.txt* document to 14.07 Frequency Analysis project in the Mod 14 Assignment folder.
2. Your frequency analysis program should also work on this file, but be sure to change the names of the input and output files in your code to *ciphertext.txt* and *ciphertextfreq.txt*, respectively.
3. When you run the program again, the frequency analysis of the letters in the *ciphertext.txt* file should be printed neatly to the screen. Copy and paste the results into a word processor and print a hardcopy of the results. Be sure to label these results as the Ciphertext Frequency Analysis to avoid confusion.

Part 4: Decipher the Message

1. Place the printouts of the frequency analysis of the Plaintext and the Ciphertext side-by-side. This is your decryption key.
2. Earlier you downloaded an encrypted message in the secretmessage.txt file. Print the message so you can begin the deciphering process.
3. The easiest way to decipher the message is to start with small one or two letter words. There are very few possibilities for one- and two-letter word combinations, so it will quickly be obvious if you are on the right track.
4. Pick a letter in an encrypted word and find the letter in the Ciphertext Frequency Analysis table printed earlier.
5. Look for the same or a similar frequency of a letter in the Plaintext Frequency Analysis table. You may find one or more exact matches or close matches with plaintext letters. Try substituting a corresponding plaintext letter for a ciphertext letter in the secret message. Keep in mind that your first choice(s) may not be correct. This is where the fun begins!
6. Use context to help you decide if you have made a correct choice. For example, if you are looking at a one letter word and the percentages tell you that "C" matches up with "b", you probably are on the wrong track. But, if "C" matches up with the letter "a" or the letter "i", you can have more confidence that you are making progress. Try substituting "a" or "i" in other words to see if the message starts to unfold.
7. Deciphering the ciphertext will take some trial and error, so be patient.
8. Continue substituting corresponding letters until the secret message is revealed.

Part 5: Evaluate the Process

1. What difficulties did you run into writing the frequency analysis program? How did you resolve these problems?
2. How closely did the frequencies of letters in the plaintext and the ciphertext correspond? What could be done to improve the accuracy of the correspondence?
3. This assignment involved writing a program and then interpreting the results the old fashioned way. Propose an extension to this project that would allow the computer to do all the work so that you only have to evaluate whether the decoded message makes senses. You don't need to actually write another program to do this, just consider how it could be done and clearly describe your solution in a well written paragraph.

Grading: Your assignment will be graded according to the following rubric.

Grading Rubric	**Pts**
Comments include name, date, and purpose of program.	1
Program written in two separate classes.	1
OOP design effectively used.	3
Methods well designed in small functional units	4
Plaintext file correctly read.	3
Frequency analysis of plaintext generated.	6
Ciphertext file correctly read.	3
Frequency analysis of ciphertext generated.	6
Frequency analysis printed neatly to screen.	3
Key correctly created.	4
Ciphertext message correctly deciphered.	10
No compiler errors.	2
No runtime errors.	2
Evaluation questions #1 and #2 correctly answered.	4
Extension proposal (#3) thoughtfully evaluaged.	6
Thoughtful PMR included.	2
Total	60

Submission: Submit the FrequencyAnalysis.java and FrequencyAnalysisTester.java files, the answers to the evaluation questions posed above, and of course the decoded secret message as Assignment 14.07 for a grade.