CIS 3362 Homework #2: Substitution Code Breaking, Vigenere Solutions

1) First, the letter frequencies can be found using the Cryptotool on the class website.

   The most common letters and their frequencies are:

```
Y    10.0%
O    9.4%
J    8.8%
P    8.2%
S    8.2%
F    7.6%
```

   Generally, the most common letter in English is ‘E,’ meaning the ciphertext ‘Y’ might map to ‘E’ in the plaintext.

   The most common trigrams in the ciphertext are OAJ, PSD, and JSG, and OAY. By guessing that these are related in some way to the most common trigrams in English—THE, AND, and ING—it can become more clear that PSD and JSG, both with ‘S’ in the middle, imply ‘S’ maps to ‘N,’ which is supported by the frequency analysis.

   Going off the prior assumption that ‘Y’ is ‘E,’ then that means OAY is THE, and at a guess, PSD is AND and JSG is ING. This would mean that a plaintext ‘D’ and ‘G’ are unchanged when encrypted, but it would match with their comparative frequencies in the text.

   Substituting these letters results in the message:

```
-IN-ETHI-I--H--TN-TI-E-N-I-E-A-T-EA-----THI-A--IGN-ENTIA-N-
T HIDING AN-THING --T---THENE-T-NEI-I--AND-NE-----G-----A—
IN ---ETHING IH--E---A-E-EADING THI-I----A-E---HA-ED-NEAG--
D---DE-I-HE-INGTHE-A-I-----TIT-TI-N-I-HE-AND----A----NDE--
TAND-HATI-EANT--T-IA-ANDE----ANDTHE-A-TTHAT-N-EA-E--ETTE---
A--INT---A-ETHE-E-T--THE---I----D---A--E--
```

   This arrangement supports the assumption that ‘D’ and ‘G’ are correctly mapped. There are three instances of “THI-” which could correspond to THIS, and an instance each of “AN-THING” and “---ETHING” which could be the words ANYTHING and SOMETHING. The message then becomes:

```
SIN-E THIS IS SHO-TNOTI-E-N-I-E-ASTYEA-S-O-THIS ASSIGNMENT
I AM NOT HIDING ANYTHING --T-O-THENE-TONEI-I--ANDONE----YG-
O--MAY-IN SOMETHING IHO-EYO-A-E-EADING THIS I-YO-A-EYO-HA-E
DONE A GOOD-O-DE-I-HE-ING THE-ASI-S--STIT-TION -I-HE-AND--
O-A--Y-NDE-STAND-HAT IMEANT-YT-IA-ANDE--O-AND THE-A-TTHAT
ON-EA-E--ETTE-S-A--INTO--A-ETHE-ESTO-THEM--I---Y DOSOAS-E--
```

Naturally, at this point more and more words become clear, and the substitution begins to fall into place. Even the word "S--STIT-TION" is present in the text. Intuition, as well as trial and error, will eventually bring you to the answer:

```
Since this is short notice, unlike past years for this
assignment, I am not hiding anything. But for the next one,
I will, and one lucky group may win something. I hope you
are reading this. If you are, you have done a good job
deciphering the basic substitution cipher and probably
understand what I meant by trial and error, and the fact
that once a few letters fall into place, the rest of them
quickly do so as well.
```

2) The process for this question is similar to that of the previous question. First, the most common letters and their frequencies are:

```
J    9.6%
G    9.3%
L    8.9%
W    8.1%
B    7.4%
C    6.9%
```

And the most common trigrams are GKW, GWO, GCH, JCO, and so on. There are actually a lot of repeating n-grams in this text. However, only JCO and JSG, each repeated three times, begin with 'J', which is implied to map to 'E' according to the letter frequencies. Assuming this is correct, looking at the most common trigrams in English that begin with 'E,' the most frequent one is ENT. Thus, another assumption is that either JCO and JSG maps to ENT.

Seeing as the E, T, A, O, I, N frequency ordering could imply JSG is ENT, this is a fair assumption to make; however, this would mean the word THE appears at most once in the text, as there is no repeating trigram of the form "G-J," which is unlikely.

Assuming instead that JCO is ENT, it can also be assumed that OFJ is THE, and plugging this into the Cryptotool doesn't reveal too much. However, there is an n-gram which might be familiar: "----T-T-T--N." Three Ts appearing in quick succession like so, with the N at the end, could remind one of the previous appearance of "SUBSTITUTION." And this would make perfect sense because, on inspection, 'W' could map to 'S,' 'E' to 'U,' 'A' to 'B,' and 'G' to 'I.' Essentially, there aren't many words that would fit the 'mold' here, and with this, the text then becomes:

```
---IN-IIS-NO-N-STHE--THE-O----B-HI-OSO-H-HE--SBO-NIN-U---N-
E-U--TE-INB--H------O--IN-TO-I-I-E-I-HE--OTE---SO-TSO-----
E-I--O-U-ENTS-O-E-IN--E-I-INE--TH-ST-ONO--O-TI-S-N--O-ENO--
---S-EO--ES-E-I--I-EIN-E--S----TO-I-SBUTIN-N-IENTTI-ES----
```

```
E-I-S-E-E-ENE---ISTSSTU--IN---N--I--E-ENTTO-I-SO--OU-SE-S-
OU----NO----IN-IISTHE-I-STTO--ITE -BOUT --E-UEN---N---
SISTOB-E—SUBSTITUTION -I-HE-SSU-H-S THIS ONE HO-E-U----
OUUSE-HISTE--HIN-S-E--
```

This seems to make a good deal of sense, and with some confidence that the assumptions so far are correct, one can further analyze the plaintext for more information, such as "-BOUT" possibly being "ABOUT," the trigram LCS being AND, the digram XX being LL (as it cannot be SS, EE, or TT), and the n-gram GCH mapping to ING. This results in:

```
AL-INDI IS-NO-NAS THE -ATHE-O-A-AB-HILOSO-H-HE-ASBO-NIN-U-A
AND EDU-ATED IN BAGHDAD A--O-DING TO-I-I-EDIAHE--OTE ALL
SO-TSO-A-ADE-I-DO-U-ENTS-O-E-ING -EDI-INE-ATHAST-ONO--O-TI-
SAND-O-ENO-ADA-S-EO-LES-E-IALI-EIN-E--S-ALLTO-I-S BUT IN
AN-IENTTI-ESA-ADE-I-S-E-EGENE-ALISTSSTUD-ING -AN-DI--E-
ENTTO-I-SO--OU-SEAS-OUALL-NO-AL-INDI IS THE -I-STTO--ITE
ABOUT --E-UEN-- ANAL-SIS TOB-EA- SUBSTITUTION -I-HE-SSU-HAS
THIS ONE HO-E-ULL—OU USED HIS TEA-HINGS -ELL
```

At this point, more and more blank spaces can be filled in, such as "-HILOSO-H-Y," "EDU-ATED," and so on. Again, intuition as well and trial and error will eventually deliver you to the decrypted message:

```
Al Kindi is known as the father of Arab philosophy. He was
born in Kufa and educated in Baghdad. According to
Wikipedia, he wrote all sorts of academic documents
covering medicine, math, astronomy, optics, and more.
Nowadays, people specialize in very small topics, but in
ancient times, academics were generalists studying many
different topics. Of course, as you all know, Al Kindi is
the first to write about frequency analysis to break
substitution ciphers such as this one. Hopefully you used
his teachings well.
```

3) I created a program to calculate the ICs, **hw2_3.c**, for each of the above ciphertexts, resulting in the following output:

```
For the first ciphertext:
IC as fraction: 3286/54285
IC as decimal: 0.060532

For the second ciphertext:
IC as fraction: 5179/87153
IC as decimal: 0.059424
```

This shouldn't be surprising, as a transposition cipher or simple substitution cipher (as in the above questions) should result in ciphertext with an IC very close to that of the language

in question, which in this case is 0.067. A polyalphabetic cipher, such as the Vigenere cipher, results in a far lower IC.

This is because the index of coincidence is defined as the likelihood that two randomly chosen letters in a group are the same, and in a transposition or simple substitution cipher, the characters may be rearranged or represented differently, but their frequencies are still the same. Thus, the index of coincidence can be a helpful tool for determining what type of cipher was applied to a plaintext.

4) First, I created a program, **vigIC.c**, to calculate the index of coincidence for each keyword length *k* from k = 1 to k = 15, which are values bounded by the possible keywords given. You can assume from the "aaaa" in the ciphertext that the key is longer than 1 character, however, as a single-character keyword in the Vigenere cipher is equivalent to a simple shift cipher, and it's extremely unlikely for any character to appear four times in a row.

   Calculating the index of coincidence in this case involves separating characters into bins based on their position in the text and the size of the keyword, *k*. So for instance, if *k* was assumed to be 4, and c_i represented the ciphertext characters, then c_0, c_4, c_8, … would go in one bin, c_1, c_5, c_9, … in another, and so on. Each bin needs to have its own IC calculated. The average of these ICs represents the overall index of coincidence for the value *k*.

   The program outputted the following ICs:

k	**IC**
1	0.040850
2	0.046332
3	0.040473
4	0.048863
5	0.048444
6	0.045620
7	0.041107
8	0.046429
9	0.046045
10	0.062133
11	0.036716
12	0.043442
13	0.041060
14	0.048926
15	0.046316

   This strongly implies that k = 10, as this is the only IC which is close to the previously mentioned norm of 0.067 for English. Knowing this, only a small portion of the potential keywords need to be tested (there are only 24 ten-letter words in the list).

   The time difference is small in this case, and a bruteforce program would find the correct key quickly even when attempting all 1000 keywords, but if the list of keys was far larger

or completely unknown, this kind of test could prove useful in narrowing down the possibilities.

To understand how to decrypt each character, it must be understood that:

$$c_i = (p_i + k_i) \bmod 26$$

where c_i, p_i, and k_i are ciphertext, plaintext, and keyword characters respectively. This means that each ciphertext character is calculated by adding together the plaintext character and the corresponding character in the keyword. Thus, the plaintext character can be found by subtracting the keyword character from the ciphertext, and so:

$$p_i = (c_i - k_i) \bmod 26$$

It must be remembered that a negative value *n* mod 26 is equivalent to *n* + 26. Applying this formula will yield the plaintext.

Combing through the plaintext results for the word “fast” reveals the key to be “management,” and the message is as follows:

```
I am testing your coding skills here to see if you can
automate reading in a list of potential keywords, apply
them to the ciphertext, and then automatically scan the
ciphertext for a particular word. The longer this
ciphertext is, the slower your program will run, but
something like this will still be very fast because I have
given you so few words to try.
```

A brute force solution has one step instead of two. Simply load the ciphertext into your program, then, read in each word from the dictionary (1000 given words), one by one. Go ahead and decrypt using each word as the key. Then, take this decryption and search for the substring "fast". If it's contained in the decrypted text, output it. When running a program that does this, only one potential plaintext is created, the one above. The implementation of this is attached in the file, **h2q4.java**. Note the ease with which we can use Java's built in methods to solve a problem like this where the number of potential keys and the size of the ciphertext are fairly small.