Final Project — Phase Two

Max Olivier

March 8, 2009

Readings
For this phase of the final project please read pages 35-38 (until example 1.22) in Lewand.
Main Ideas

- Up until now we have only discussed how to decode message when we have the key. But what if we don't? This is obviously the more realistic and also more interesting case. This is the cryptanalysis part of the cryptology. The reading for this phase of the final project will show you one technique called *frequency analysis* that suggests how to go about decrypting messages when we do not have the key. So by the end of this phase you should be familiar with frequency analysis and how it can help in decoding the three ciphers we have been using this term.

Exercise 1 (Cryptololgy Program)
Add a decode_rand() function to your Add_Cipher.cpp program that takes a pointer to a string as a parameter and decodes that string (though the returns type as with our other encode and decode functions should be void). A few things to keep in mind before you get started:

- You are going to want to find the most frequently occurring letter in the encoded string, so you will need a data structure that can record all of the letters in the string, how many times they occur, and return the most frequently occurring one. Does this sound familiar at all ...? And for those of you wondering whether an array could have accomplished the same task, the answer is yes and it would have been almost as efficient, but then you would not have had the opportunity to make a linked list. Thus, I would suggest you use your linked list for this part of the program. Before you do so, however, you will need to convert your LinkedList.cpp program into a header file so that it can be included in the new program with an include directive. This will not be difficult (just save as .h instead of .cpp), but remember to delete the main function in the header file.
- You will also need to know the most frequently appearing letters in the alphabet. To find them either use the table either use the table in Lewnd or wikipedia. As for storing them in your program, I would suggest you store them in a string similar to the way the Translator.h file stores the alphabet string, but in order of frequency in the alphabet.
- You will need to convert the most frequently appearing character into its corresponding number value. You will find the get_index() function from your translator helpful in this.
- Remember that possible keys (and therefore decodings) may be wrong, and therefore before committing to a decoded message you should check with the user that the decode message is the original. If not, you should keep searching.
- You will want to use the regular decode function to test decodings with possible keys.

- Make sure the search doesn't get stuck in an infinite loop.
- Make sure that you don't alter the original string until the very end when you are sure the decoding is right. Otherwise if you alter it and the decoding is wrong you will not have the original encoded message to decode the next time.
- Finally, all the while we will be assuming that whoever encrypted the message used a proper key and kept to our usual string types of all lower case letters and no punctuation or other symbols.