CS 280
Fall 2019
Programming Assignment 1

Part 1 due 9/25
Part 2 due 10/2

In cryptography, we use encryption ciphers to encode ("encrypt") a plaintext into a ciphertext. In theory the ciphertext is only readable by someone who knows how the plaintext was encrypted. If they have this information they can "decrypt" and recover the plaintext.

There are some very simple ciphers, known as "substitution ciphers". They're so simple that they're not used any more because they are easy to break… but they are great for us because they are easy to program!

First is the "rot13" cipher. Rot13 rotates each letter 13 positions in the alphabet. So an a becomes an n, and an o becomes a b. (the rotation wraps). If the rot13 cipher is applied a second time, the original plantext is restored: n becomes a, b becomes o.

Obviously the rot13 cipher provides no security! It's really used in various places on the internet to hide joke punch-lines or spoilers.

Another cipher, the "caesar cipher", is a substitution cipher with a little more security. The cipher uses a dictionary where each plaintext letter has a corresponding ciphertext letter paired with it. For example we might say a is substituted with m, b is substituted with x, etc. to encrypt, each letter is replaced with its ciphertext letter. To decrypt, the reverse process takes place: each ciphertext letter is replaced with the plaintext letter.

This cipher is not particularly strong. It is quite easy to guess the plaintext-ciphertext mappings by exploiting knowledge of letter frequency and word length. It is, however, harder to break than rot13.

For this assignment, we will write a C++ program capable of doing 4 things:

1. Perform a rot13 substitution
2. Perform a caesar encryption given a dictionary
3. Perform a caesar decryption given a dictionary
4. Create a random caesar cipher dictionary

The format for the caesar cipher dictionary is a file with 26 pairs of letters, one per letter of the alphabet, in alphabetical order. Each pair has two letters: the plaintext letter and the ciphertext letter. The pairs are separated by whitespace. While the dictionary contains only lowercase letters, the mappings also apply for uppercase letters. Note that the ciphertext letters must be

unique; you cannot have two different plaintext letters that map to the same ciphertext letter. **You also cannot have a plaintext letter map to itself.**

For all of the substitution operations, all upper and lower case letters on input should be substituted (or rotated as the case may be) before being output. All other characters are simply copied unchanged from input to output.

The program takes several command line arguments. The first, which is required, tells the program what operation to perform:

-r	perform rot13 substitution.
-g	generate a random caesar cipher dictionary
-e	encrypt using the caesar cipher
-d	decrypt using the caesar cipher

If the first argument is missing, the program should print MISSING COMMAND, then stop.

If the first argument is not one of the four listed above, print the first argument followed by a space and NOT A VALID COMMAND, then stop.

For -r, there is an optional second argument. If provided, it is the name of the file to read from. If it is not provided, the program should read from standard input. The output should be generated to the standard output. If a filename is provided but the file cannot open for any reason, the program should print the filename followed by a space and FILE COULD NOT BE OPENED, then stop.

For -g, the dictionary should be printed to the standard output. You must ensure that each plaintext letter maps to a unique ciphertext letter, **and that no letter maps to itself.**

For both -e and -d there is a required second argument, which is the filename of the dictionary. If the second argument is missing, print the message NO DICTIONARY GIVEN, and stop. If the dictionary cannot open for any reason, the program should print the filename followed by a space and DICTIONARY COULD NOT BE OPENED, then stop.When reading the dictionary, you must ensure that each plaintext letter maps to a unique ciphertext letter. **If you find a case where the dictionary does not contain a two-letter pair, print FORMATTING ERROR followed by a space and the entry that you read, and stop. If you find a case where a ciphertext is duplicated, the program should print DUPLICATE CIPHERTEXT L, where L is the duplicated letter, and stop. If you find a plaintext that maps to the same letter in ciphertext, the program should print MAPPING ERROR L, where L is the plaintext letter,**

and stop. If you find a case where the dictionary is not in alphabetical order, you must print MISSING LETTER L, where L is the missing letter, and stop.

Both -e and -d support an optional third argument, which is the file to read from. If it is not provided, the program should read from standard input. The output should be generated to the standard output. If a filename is provided but the file cannot open for any reason, the program should print the filename followed by a space and FILE COULD NOT BE OPENED, then stop.

In all cases if there are too many command line arguments, your program should print TOO MANY ARGUMENTS, then stop.

The test cases for part 1 will cover -r and -g. Cases for -e and -d will be added for part 2.