CMPUT 299, Winter 2018, Assignment 2

Acknowledge **all** *resources consulted (discussions, texts, urls, etc.) in working on this assignment.* **Without this acknowledgement, your assignment will not be graded.** *Non-detailed oral discussion with others (including non-students) is permitted as long as any such discussion is summarized and acknowledged by all parties; the viewing or exchanging of any written work, even in rough or preliminary form, is expressly forbidden, as is any detailed discussion.* **Late assignments will not be accepted.**

The Caesar Cipher does not offer much security, as the number of keys is sufficiently small to allow a brute-force approach to cryptanalysis. However, with some modifications, it can be greatly improved. In this exercise, you will modify the given caesarCipher.py code, which implements the Caesar Cipher, to produce an implementation of a much stronger cipher.

Some problems in this assignment take the form of short-answer questions. Your answers should be submitted as a PDF named "a2.pdf", with the answers to the short-answer problems presented in order, with clear demarcations indicating where your response to each problem begins. Your answers should not be longer than 100 words each.

You will produce four files for this assignment: three Python files (one for each of problems 1, 2, and 4, named "a2p1.py", "a2p2.py", and "a2p4.py" respectively), and your a2.pdf, containing your responses to the short-answer questions, problems 3 and 5. For your submission, put all four of these files in a directory called 299-as-2. Zip this directory so that you have a new file called 299-as-2.zip. To submit the assignment, upload your zipped assignment to the assignment page in eClass.

Problem 1 Make a copy of the given "caesarCipher.py" called "a2p1.py", and modify it so that it takes the key, mode ('encrypt' or 'decrypt'), and message as command line arguments, in that order. For example,

```
python3 a2p1.py 13 encrypt 'encrypted with a shift of thirteen'
RAPELCGRQ JVGU N FUVSG BS GUVEGRRA

python3 a2p1.py 13 decrypt 'RAPELCGRQ JVGU N FUVSG BS GUVEGRRA'
ENCRYPTED WITH A SHIFT OF THIRTEEN
```

You can assume the following:

- The key will be an integer between 0 and 25 (inclusive).
- The mode will be either *encrypt* or *decrypt*; otherwise, the program should terminate.
- The message (plaintext when encrypting or ciphertext when decrypting) will be enclosed in single quotes.

Problem 2 The weakness of the Caesar Cipher comes from the fact that every letter is enciphered the same way. If the first letter of the message is shifted forward by three positions, *every* letter is shifted forward by three positions. Let's start by fixing this flaw. Make a copy of your "a2p1.py" code named "a2p2.py" and modify it such that:

1. The first letter of the message is shifted according to the chosen key, exactly as before (i.e. a key of '3' shifts the first letter up by 3 for encryption).
2. All remaining letters are shifted according to the previous letter of the message. So, if the message is simply the word "gold", the 'g' is enciphered as per the Caesar Cipher, but the 'o' is shifted up 6 positions, since the letter 'g' corresponds to the number 6 (remember that 'a' is 0, 'b' is 1, and so on until 'z' is 25). The 'l' is then shifted up 14 positions, and the 'd' is shifted up 11 positions. If the initial key is '3', this gives a ciphertext of "juzo".

Your implementation should still be able to encrypt *or* decrypt, depending on the "mode" variable.

Problem 3 Short answer: Is the cipher you implemented in problem 2 more secure than the Caesar Cipher? Why or why not? Include your response in your a2.pdf.

Problem 4 Now make a copy of your "a2p2.py" code, name it "a2p4.py", and modify it so that the key is not a single letter or number, but a "word". For this exercise, a "word" is simply any string of one or more letters; it need not be meaningful in any language. So, "cdgd" is considered a word here. We will call this word the *keyword*. Modify the code so that it enciphers a message as follows:

1. Let's call the length of the keyword n (so if the keyword is "cdgd", n is 4. The first n letters of the message are shifted according to the n^{th} letter of the keyword. So, if the keyword is "cdgd", the first letter of the message would be shifted by 2 (since 'c' corresponds to 2), the second would be shifted by 3, the third by 6, and the fourth by 3.
2. If there are more than n letters is the message, then the m^{th} letter, where $m > n$, is shifted according to the $(m-n)^{th}$ letter of the plaintext message itself. So, if the message is "helloworld" and the keyword is, again, "cdgd", the 'o', being the fifth letter, is shifted according to letter $(5-4) = 1$ of the message, which is 'h', which gives a shift of 7, so the 'o' is enciphered as 'v'. The sixth letter, 'w', is enciphered using the next letter in the plaintext, which is 'e', so the 'w' is enciphered as 'a'. This process ultimately gives a ciphertext of "JHROVAZCZZ".

Non-letter characters should be skipped for both encryption and decryption; they should not be changed, nor used as part of the key; rather, they should be appended to the output "as is", and the encryption/decryption process should then proceed as if the non-letter character was not there. So, encrypting "hello, world" with key "cdgd" would give a ciphertext of "JHROV, AZCZZ", as the comma and space characters are not enciphered. Again, your code must be able to both encrypt and decrypt, depending on the "mode" variable.

Problem 5 Short answer: While the cipher you implemented in Problem 4 is far from being the strongest cipher we will consider, it is much stronger than the Caesar Cipher we stared with. Explain briefly why this is. Include your response in your a2.pdf.