CMPUT 396, Fall 2019, Assignment 2

Before beginning work on this assignment, carefully read the Assignment Submission Specifications posted on eClass.

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 396-as-2. Zip this directory so that you have a new file called 396-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.
```

The key will determine the shift for all alphabetic characters. Non-alphabetic characters are **NOT** to be shifted, but instead simply appended to translated message. All uppercase characters are to remain uppercase, and similarly for lowercase. For example from the above, `E` is shifted by 13 to become `R`, while `e` is shifted by 13 to become `r`.

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 (Note that these quotes will not be a part of the string in Python).
- The message will only contain ASCII characters (Non-alphabetic characters are treated separately, see below).

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 and the sum of all previous shifts. Using the example from **Problem 1** for the word `Encrypted`, `E` is shifted by 13 (the key) to the letter `R`. `n` is shifted by $4 + 13 = 17$ which is the index of `E` plus the sum of previous shifts 13. `c` is shifted by $13 + 17 = 30$ which is the index of `n` plus the sum of previous shifts 17. This process is continued.

```
python3 a2p2.py 13 encrypt 'Encrypted with a Shift of Thirteen.'
Regxvkdhk goho o Gnvat hm Fmuleimz.

python3 a2p2.py 13 decrypt 'Regxvkdhk goho o Gnvat hm Fmuleimz.'
Encrypted with a Shift of Thirteen.
```

Your implementation should still be able to encrypt *or* decrypt, depending on the "mode" variable, similar to **Problem 1**.

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, "dxjc" 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 n be the length of the keyword, and suppose the keyword is "dxjc" (so here $n = 4$). The first n letters of the message are shifted according to the n^{th} letter of the keyword, plus the sum of the previous shifts. Using the prevous example for the word `Encrypted`, `E` is shifted by 3 (since 'd' corresponds to 3) to the letter `H`. `n` is shifted by $23 + 3 = 26 = 0$ (since 'x' corresponds to 23 and the sum of previous shifts is 3) to the letter `n`. `c` is shifted by $9 + 26 = 35 = 9$ (since 'j' corresponds to 9 and the sum of previous shifts is 26) to the letter `L`. Similarly, `r` is shifted by $2 + 35 = 37 = 11$ to `c`.
2. If there are more than n letters in the message, then the m^{th} letter, for $m > n$, is shifted according to the $(m - n)^{th}$ letter of the key. Continuing with the above example, `y` is the 5^{th} letter, but there are only 4 letters in the key, so it will be shifted by letter $(5 - 4) = 1$ of the key, plus the previous sum of shifts as before. `y` is shifted by $3 + 37 = 40 = 14$ (since 'd' corresponds to 3 and the sum of previous shifts is 37) to the letter `m`.

```
python3 a2p4.py dxjc encrypt 'Encrypted with a Shift of Thirteen.'
Hnlcmanac snar h Izdxu rl Wtwihbdp.

python3 a2p4.py dxjc decrypt 'Hnlcmanac snar h Izdxu rl Wtwihbdp.'
Encrypted with a Shift of Thirteen.
```

For problems 1, 2, and 4, 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 "dxjc" would result in the following call:

```
python3 a2p4.py dxjc encrypt 'Hello, world.'
Keuwc, hinkz.

python3 a2p4.py dxjc decrypt 'Keuwc, hinkz.'
Hello, world.
```

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.