CMPUT 396, Fall 2019, Assignment 1

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

As noted in chapter 8, the number of possible keys for the transposition cipher is equal to half the length of the message. This means that, unless the message is extremely long, it is still possible to use the brute force technique to decipher a message enciphered with the transposition cipher, even without the key. This exercise will guide you to implementing a stronger cipher, much more resistant to brute force attacks.

The transposition cipher enciphers a message by writing the message down in rows, and then reading of the columns, from left to right. For example, to encipher the message
"CIPHERS ARE FUN" with the key, 5, we create the following table. We then read the columns left to right to get the ciphertext "CREIS P FHAUERN"

```
C I P H E
R S _ A R
E _ F U N
```

Instead of always reading the columns left to right, suppose we read the columns of in the order 4, 5, 3, 1, 2. We would get the ciphertext "HAUERNP FCREIS " (note the trailing space character). On the other hand, if we choose the order 2, 4, 1, 5, 3, we get the ciphertext "IS HAUCREERNP F". In fact, the number of distinct orders in which we can read these five columns is 5! (120). Even if a hacker knows we used five columns in our table, they still wouldn't know which of the 120 possible orders will allow them to break the code.

Better still, the number of possible order increases very quickly with the number of columns, since the factorial function grows extremely fast. If we used ten columns instead of five, the number of possible orders is over three million. If we used fifteen, the number of orders is over one trillion. The brute force technique will fail as long as the number of columns is not too small.

Note that the cipher's key is no longer a single number, but a particular permutation of the numbers from 1 up to a particular number. For example, "2, 4, 1, 5, 3" is a possible key to this cipher. This modification – changing the key from a number to a permutation, and reading the columns in that order– results in the *columnar transposition cipher*.

For your implementation, use TRANSPOSITIONENCRYPT.PY and TRANSPOSITIONDECRYPT.PY from the textbook source files as a starting point.

You will produce five files for this assignment: four Python scripts (one for each of the four problems), and, for problem four, a file containing a decipherment of an enciphered text extracted from Wikipedia.

Problem 1: Modify transpositionEncrypt.py so that it implements encryption using the columnar transposition cipher. Name the resulting Python file "a1p1.py". The "encryptMessage" function takes two parameters in this order: key, a list of integers which determine the order in which the columns are to be rearranged, and message, a string. Every integer in key is unique and is between 1 and the number of columns. For example, encryptMessage([2,4,1,5,3], 'CIPHERS ARE FUN') should return 'IS HAUCREERNP F'.

If the length of the given message is not a multiple of the number of columns, your program should use the shaded box technique described in the textbook. For example, encryptMessage([1, 3, 2], "ABCDEFG") should return 'ADGCFBE'.

Problem 2: Now that you can encrypt using the columnar transposition cipher, the next task is to be able to automatically decrypt messages. Modify transpositionDecrypt.py so that it implements decryption using the columnar transposition cipher. Name the file containing your code "a1p2.py".

To test your code, try decrypting the ciphertext
"XOV EK HLYR NUCO HEEEWADCRETL CEEOACT KD", which was encrypted using the key [2,4,6,8,10,1,3,5,7,9].

This program must should be able to decrypt messages that were encrypted using the program from problem one.

Problem 3: At this point, you have Python modules which should be able to encrypt and decrypt with the columnar transposition cipher. The next step is to automate the testing of these modules.

Modify transpositionTest.py so that it tests the modules you wrote in problems 1 and 2. Name the file containing your code "a1p3.py".

The program should run tests on 5 random strings, generated the same way, and should test all valid key lengths (key lengths which divide the message length). For each key length, 5 random keys should be generated and tested. Your program must print each key that it tests.

Problem 4: Modify transpositionFilecipher.py to use the columnar transposition cipher. Name the file containing your code "a1p4.py". After testing your program, use it to decrypt the file "mystery.txt", which contains some text extracted from Wikipedia, encrypted using the columnar transposition cipher with the key [2,4,6,8,10,1,3,5,7,9]. The output should be saved in a file named "mystery.dec.txt", which should be included in your submission.