Mini Project 1

(due on Sept. 19, 2023, end of the day)

Task 1: In this mini project, you are first asked to implement a simple Vigenere Cipher.

- The algorithm for encryption is: $E_K(m) = m + K \bmod 26$
- The algorithm for decryption is: $D_K(m) = m - K \bmod 26$

Format: (1) the plaintext/ciphertext should only contain letters (but you do not need to check the validity of the input. We assume that the input is always valid). (2) Spaces in the plaintext should be removed. (3) Your input/output should be text strings, with both uppercase and lowercase letters. The same letter in upper and lower cases are treated as the same. That is, both "A" and "a" must be converted to "1" (or "0") in your program.

Task 2: Next, you are expected to implement a brute force password cracker based on the Vigenere Cipher you just implemented. Your password cracker is expected to take three parameters: (1) a string of ciphertext; (2) an integer keyLength that denotes the length of the key; and (3) an integer firstWordLength that denotes the length of the first word of the plaintext.

Your password cracker will test every possible key that has the length of keyLength: from all "A"s to all "Z"s. You cannot exploit the dictionary to guess the key, since the key may not be a valid word.

For each key candidate, you will generate a "plaintext", and compare it with the dictionary (provided to you). In particular, you only need to check if the first word (number of letters of the word is given in firstWordLength) is a valid word in the dictionary. To do this, you need to load the dictionary into memory before processing any key, and search if the first word of the "plaintext" is in the dictionary. If Yes, display the plaintext and the key. However, do not stop, as the "plaintext" might be wrong.

Efficiency is very important in evaluating each "plaintext" candidate. In some cases, a wrong key may generate a valid first word. Hence, you may get several "plaintexts" after all possible keys are tested. This is acceptable. You can look at the outputs and determine which key is correct.

Task 3: Use the brute force password cracker for the Vigenere Cipher you implemented in Task 2 to decrypt the following messages.

1. "MSOKKJCOSXOEEKDTOSLGFWCMCHSUSGX"
   key length = 2; firstWordLength = 6

2. "PSPDYLOAFSGFREQKKPOERNIYVSDZSUOVGXSRRIPWERDIPCFSDIQZIASEJVCGXAYBGYXFPSREKFMEX
   EBIYDGFKREOWGXEQSXSKXGYRRRVMEKFFIPIWJSKFDJMBGCC"
   keyLength=3; firstWordLength = 7

3. "MTZHZEOQKASVBDOWMWMKMNYIIHVWPEXJA"
   keyLength=4; firstWordLength = 10

4. "SQLIMXEEKSXMDOSBITOTYVECRDXSCRURZYPOHRG"
   keyLength=5; firstWordLength = 11

5. "LDWMEKPOPSWNOAVBIDHIPCEWAETYRVOAUPSINOVDIEDHCDSELHCCPVHRPOHZUSERSFS"
   keyLength=6; firstWordLength = 9

6. "VVVLZWWPBWHZDKBTXLDCGOTGTGRWAQWZSDHEMXLBELUMO"
   keyLength=7; firstWordLength = 13

When key length increases, cracking becomes slow. Please optimize your code, and at least finish the first 5. Please record the time needed to decrypt each message.

Please submit your code and a short report. The report shall demonstrate the plaintexts that you discovered. The report shall also discuss the efficiency of password cracking.