Introduction to Cryptography
Project 2

(1) Write a function in Python that
   (a) accepts a message (string), a multiplicative key (integer), and boolean variable as arguments;
   (b) checks that the key is valid (mod 26);
   (c) performs the multiplicative encryption to each letter modulo 26 if the boolean variable is TRUE and performs the multiplicative decryption to each letter modulo 26 if the boolean variable is FALSE;
   (d) returns the encrypted/decrypted message as a string.

(2) Write a function in Python that
   (a) accepts an integer message, multiplicative key (integer), a modulus n (integer), and a boolean variable as arguments;
   (b) checks that the key is valid (mod n);
   (c) checks that the modulus is larger than the integer;
   (d) performs the multiplicative encryption to the integer message modulo n if the boolean variable is TRUE and performs the multiplicative decryption to the integer message modulo n if the boolean variable is FALSE;
   (e) prints the encrypted/decrypted integer message.

(3) (a) Show that 37 has a multiplicative inverse modulo 2798989898
   (b) Find the multiplicative inverse of 37 modulo 2798989898
   (c) Decrypt the following ciphertext message, which was encrypted by performing an additive shift on each letter, converting the plaintext message into five-letter blocks, and then multiplying each block by 37 modulo 2798989898.

1811758588 853934819 2412335942

(4) The **Affine Cipher** is an encryption scheme which combines the Additive and Multiplicative Caesar ciphers. To encrypt a plaintext message with an affine cipher, we need an additive key A, a multiplicative key M, and a modulus n. Then given a character *char* in the message, we encrypt *char* by computing $M * char + A \mod n$.
   Write a function in Python that
   (a) accepts a lowercase message (string), a multiplicative key (integer), an additive key (integer) and a boolean variable as arguments;
   (b) checks that the keys are valid (mod 26);
   (c) performs the affine encryption to each letter modulo 26 if the boolean variable is TRUE and performs the affine decryption to each letter modulo 26 if the boolean variable is FALSE;
   (d) returns the encrypted/decrypted message as a string.

(5) Recall that Euler's Totient Function, $\phi(n)$ computes the number positive integers less than n which are relatively prime to n. We observed in a previous assignment that if p and q are distinct prime numbers, then $\phi(p) = p - 1$ and $\phi(pq) = (p-1)(q-1)$. Suppose that you are given a large integer n, and you know that it is factorable as $n = pq$, but you are not given the distinct prime numbers p and q. Can you compute $\phi(pq)$? Why or why not?