Euler's totient and RSA

(exercises)

Application of discrete models

1. Calculate the elements of $\mathbb{Z}_m^*$ for a given $m \in \mathbb{N}^+$ modulus. Print them from $m = 1$ to 20.

2. Write a function that calculates $\varphi(n)$
   a) using simple counting; b) from the prime factorization of n.

3. Calculate manually:
   a) $\varphi(25)$; b) $\varphi(27)$; c) $\varphi(30)$; d) $\varphi(36)$; e) $\varphi(45)$; f) $\varphi(81)$; g) $\varphi(100)$; h) $\varphi(600)$.

4. Write a function that calculates the modular power $a^n \bmod m$ using Euler's totient theorem, where $a \in \mathbb{Z}$, $n, m \in \mathbb{N}^+$ and $\gcd(a, m) = 1$.

5. a) Generate an RSA key pair. For simplicity, the following (insecure) setup can be used: prime numbers can be generated using `random_prime()` (e.g. `random_prime(1000)`), and the public exponent can be the first odd number which satisfies the criteria of RSA.
   b) Encrypt a message m ($0 \leq m < n$) using the public key.
   c) Decrypt the ciphertext c using the private key, i.e. restore the original m.

6. Write a program that tries to break RSA using the `factor()` function, i.e. it tries to solve $m^e \equiv c \pmod{n}$ for m using only the public key. Break the encrypted message from the previous exercise.

7. Speed up the implementation of RSA using the chinese remainder theorem.

8. a) Write a function that checks whether a given $n \in \mathbb{N}^+$ is a prime number using the Fermat primality test. (Use a single random base a.)
   b) Find an n for which the test gives an incorrect result.
   c) Improve the test by repeating it k times (using separate random a's in each step), where k is also an input parameter.

9. Find all Fermat pseudoprimes to base $a = 2$ between 1 and 10000.

10. Find all Carmichael numbers between 1 and 10000.

11. Implement the Miller–Rabin primality test.

12. Find all strong pseudoprimes to base $a = 2$ between 1 and 10000.