Public Key Cryptography **Algebraic Structures**

1. Use the Python command `pow(3,100,8)` to find $3^{100} \pmod 8$. You can also use `pow(3,-1,8)` to find the inverse of 3 in $\mathcal{U}(8)$.

2. What does Python say if you try to find the inverse of 2 in $\mathcal{U}(8)$?

3. The `sympy` library has a function to compute the Euler phi-function called `totient`. Import `totient` from `sympy` and test it using the command `totient(8)` to find $\phi(8)$. How many elements are in $\mathcal{U}(8)$? What are they?

4. Another function in `sympy` lets us generate random prime numbers. Use the command `sympy.randprime(10000,20000)` to find a random prime number between 10,000 and 20,000. Record what you get as p. Then generate a second random prime number q. Let $n = pq$ and $m = \phi(n)$.

5. The number n is half of your public key. To get the other half, choose any number E that is relatively prime to m (i.e., $\gcd(m, E) = 1$). Note that `sympy` also has a `gcd` function. The pair n and E are your public keys.

6. Use the Python command `D = pow(E,-1,m)` to find the inverse of E in $\mathcal{U}(m)$. Keep D and m secret, so that only you can decode messages. They are your private keys.

7. Once you publish E and n, anyone can encrypt a message x by calculating $y = x^E \pmod n$. Only you will be able to decrypt y by calculating $y^D \pmod n$. Try encrypting a number x and then decrypting the result using these two formulas.