COMP 4958: Lab 1

Put your two functions in a module named `Lab1`. Your file should be named `lab1.ex`. Maximum score: 10

Implement the following 2 functions which can be used in the RSA cryptosystem. (The RSA cryptosystem uses different keys for encryption and decryption of messages.)

1. Given two positive integers a and n. The multiplicative inverse of a modulo n, if it exists, is a positive integer t, less than n, satisfying $a \times t \equiv 1 \bmod n$.

   The following is the pseudocode for an algorithm to find the multiplicative inverse of a modulo n:

```
function mod_inverse(a, n)
    t := 0;     newt := 1
    r := n;     newr := a

    while newr != 0 do
        quotient := r div newr
        (t, newt) := (newt, t - quotient * newt)
        (r, newr) := (newr, r - quotient * newr)

    if r > 1 then
        return :not_invertible
    if t < 0 then
        t := t + n

    return t
```

   Implement the above algorithm in Elixir in a function named `mod_inverse(a, n)`. It returns the multiplicative inverse if it exists, or the atom `:not_invertible` if it does not.

2. Assume a, m and n are positive integers. Implement a function `mod_pow(a, m, n)` that returns $a^m \bmod n$, i.e., the remainder of a^m when divided by n. Your function should be fast even for large values of m. Use the fact that $(x \times y) \bmod n = ((x \bmod n) \times (y \bmod n)) \bmod n$ to keep the numbers in your program "small".

Some information about the RSA cryptosystem:

In RSA, the modulus n is the product of 2 large primes p and q. A number e relatively prime to $(p-1) \times (q-1)$ is chosen. (According to Wikipedia, a common value for e is 65537.)

- The pair $\{e, n\}$ is the public key
- The corresponding private key is the pair $\{d, n\}$, where d is the multiplicative inverse of e modulo $(p-1) \times (q-1)$. (d can be calculated using the first function above.)
- The sender of a message M, represented by a large number, uses the public key pair to encrypt it to $C = M^e \bmod n$. (This is where the second function above comes in. Note that M should be less than n.)
- The recipient of the encrypted message C can recover M using the corresponding private key pair via $M = C^d \bmod n$. (Again using the second function above.)

You may be interested in experimenting with this.