ISEC 2000 Fundamental Concepts of Cryptography
Lab 6
@ Computing, Curtin University

Notes:

- Make sure you complete prac questions. Assignments are highly related to them. In fact, programming questions are building blocks for the assignments.
- Group discussion is encouraged but make sure you complete questions individually.
- Ask questions not just answers. There is no answer of prac questions documented or provided for computing units.
- You can use your preferred programming language, C/C++, Java, or Python. Do NOT rely on libraries excessively.

1. Perform encryption and decryption using the RSA algorithm for the following: $p = 5$, $q = 13$, $e = 5$, $m = 8$.

2. In a public-key system using RSA, you intercept the ciphertext $c = 20$ sent to a user whose public key is $e = 13$, $n = 77$. What is the plaintext m?

3. Modular exponentiation is a type of exponentiation performed over a modulus. It is very important in the field of public-key cryptography. It concerns the calculation of $y = x^H \, mod \, n$.

   Square-and-Multiply, aka left-to-right binary method, is one of the most efficient methods to calculate modular exponentiation. The following is the pseudocode of the algorithm.

Algorithm 1 Left-to-Right Binary Method

Input: Exponent H, base element x, modulus n
Output: $y = x^H \, mod \, n$

1: Determine binary representation $H = (h_t, h_{t-1}, \cdots, h_0)_2$
2: Initialise $y := 1$
3: **FOR** $i = t$ to 0
4: $\quad y = y^2 \, mod \, n$
5: $\quad$ **IF** $h_i = 1$ **THEN**
6: $\quad\quad y = y * x \, mod \, n$
7: **return** y

   Similarly, right-to-left method as follows.

```
function modular_pow(x, H, n) is
    if n = 1 then
        return 0
    y := 1
```

```
x := x mod n
while H > 0 do
    if (H mod 2 == 1) then
        y := (y * x) mod n
    H := H >> 1
    x := (x * x) mod n
return y
```

Implement either binary modular exponentiation algorithm and use it to calculate $y = x^H \, mod \, n$ with some big big big numbers! (Note: you should also replace the calculation of $r = a^{\frac{p-1}{2}} \, mod \, p$ in your Lehmann algorithm using this algorithm).