CS 3510 - Honors Algorithms
Homework 7 - Solutions

1. (Least Common Multiple) One important observation to make is that the LCM of two integers a and b satisfies $\text{lcm}(a,b) = ab/\gcd(a,b)$. This can be shown by the fact that a and b can be factorized in a unique manner as $a = \prod_{i=1}^{k} p_i^{r_i}$ and $b = \prod_{i=1}^{k} p_i^{s_i}$ where p_i are all the primes appearing in the factorization of a or b and r_i, s_i are non negative integers (some of the r_i's and s_i's may be 0). Now note that $\text{lcm}(a,b) = \prod_{i=1}^{k} p_i^{\max(r_i,s_i)}$ and $\gcd(a,b) = \prod_{i=1}^{k} p_i^{\min(r_i,s_i)}$. Trivially, $\max(r,s) + \min(r,s) = r + s$ and this implies $\text{lcm}(a,b) \cdot \gcd(a,b) = ab$ (**Verify**).

   Also, verify that the LCM of 3 numbers $\text{lcm}(a_1, a_2, a_3) = \text{lcm}(\text{lcm}(a_1, a_2), a_3)$. From this, one can inductively show that

$$\text{lcm}(a_1, a_2, \ldots, a_n) = \text{lcm}(\text{lcm}(a_1, a_2, \ldots, a_{n-1}), a_n)$$

   So we can use the following algorithm to calculate the LCM.

```
LCM(a(1),a(2),..., a(n))
     If n=1 return a(1);
     Else { L = LCM(a(1),a(2),..., a(n-1));
            return L*a(n)/gcd(L,a(n));  }
End
```

   The algorithm is correct because of the above arguments.

   If $n = 1$, we do not use any GCD computation. If $n = 2$, we need one call to the GCD function. If $n > 2$, we need one GCD function plus a recursive call to an LCM function with $n-1$ arguments. By induction, we need $n-1$ calls to the GCD algorithm. (Alternatively, you can think of it as the recurrence relation where $T(n)$ denotes the number of calls to the GCD subroutine, $T(1) = 0, T(2) = 1$ and $T(n) = T(n-1) + 1$)

2. (RSA) We have $p = 17, q = 19$. So $N = pq = 323$ and $\phi(N) = (p-1)(q-1) = 288$. The public key $e = 25$ is given. **Check** by running the Extended Euclid that $25^{-1} \equiv 265 \bmod 288$. So the public key pair is $(e, N) = (25, 323)$ and the associated private key pair

is $(d, N) = (265, 323)$. Let us encrypt the message $M = 10$. The encrypted message is $X = M^e \bmod N$.

$$\begin{aligned}
10^2 &\equiv 100 \bmod N \\
10^4 &\equiv 310 \equiv -13 \bmod N \\
10^8 &\equiv 169 \bmod N \\
10^{16} &\equiv 137 \bmod N \\
10^{24} &\equiv 10^{16} \cdot 10^8 \equiv 220 \bmod N \\
10^{25} &\equiv 10^{24} \cdot 10 \equiv 262 \bmod N
\end{aligned}$$

The encrypted message is $X = 262$. For decryption, we need to use the private key and calculate $Y = X^d \bmod N$.

$$\begin{aligned}
262^2 &\equiv 168 \bmod N \\
262^4 &\equiv 123 \bmod N \\
262^8 &\equiv 271 \bmod N \\
262^{16} &\equiv 120 \bmod N \\
262^{32} &\equiv 188 \bmod N \\
262^{64} &\equiv 137 \bmod N \\
262^{128} &\equiv 35 \bmod N \\
262^{256} &\equiv 256 \bmod N \\
262^{264} &\equiv 262^{256} \cdot 262^8 \equiv 254 \bmod N \\
262^{265} &\equiv 262^{264} \cdot 262 \equiv 10 \bmod N
\end{aligned}$$

So we can see the decryption $X^d \bmod N$ indeed gives us the original message $M = 10$.

3. (Digital Signature)

   (a) Digital signatures provide a way of authenticating the fact that it is indeed the sender who has sent the message. In eletronic communication, we would like to have an analogue to the signature in the physical communication. For example, if the letter is a letter to a bank requesting money transfer, the bank would require some means of authenticating that the letter is indeed from the person who claims to have sent the message. Thus, digital signatures provide a way of authenticating the sender, and the contents of the message as well.

   (b) Given the public key pair (N, e) and the message M, we can verify that the given signature $S = M^d$ was created by the intended sender, using the following procedure.

```
Verify ((N,e), S, M)
    If (S^e = M mod N) then return TRUE
    Else return FALSE
End
```

   The private key - public key pair (d, e) is such that e and d are multiplicative inverses modulo $\phi(N) = (p-1)(q-1)$, where N is the product of the primes p and q. The algorithm Verify checks if $(M^d)^e \equiv M \pmod{N}$. This is true since $(M^d)^e) = M^{de} = M^{ed}$. And by the proof of RSA cryptosystem, we have $M^{ed} \equiv M \pmod{N}$.

   Someone without our help cannot find out the private key d, so a false signature $T \neq M^d \bmod N$ is what one could produce otherwise. When we use the Verify algorithm, $T^e \neq M \bmod N$ would return FALSE. So the Verify algorithm would return TRUE if and only if the signature was $S = M^d \bmod N$.

4. (Breaking RSA using Signatures) The hacker's message is $Y = r^e X = r^e M^e \bmod N$. When we sign the message, as described in the last question, we have to give the signature $S = Y^d = (rM)^{ed} = rM \bmod N$. The hacker can choose the random r such that it is relatively prime to N, and can compute $r^{-1} \pmod{N}$ using extended Euclid. Then he can proceed to do $r^{-1}S = r^{-1}rM = M \bmod N$ and recovering the original message.

5. (Taking care of Carmichael Numbers)

   (a) A non trivial square root of 1 modulo s is a number $x \notin \{1, -1\}$ and $x^2 \equiv 1 \bmod s$.

   Let x be a square root of 1 modulo s. This means that $x^2 - 1 \equiv 0 \bmod s$. That is, $x^2 - 1 = (x+1)(x-1)$ is a multiple of s. Since s is a prime, either $x+1$ or $x-1$ has to be a multiple of s. This would imply $x \equiv -1 \bmod s$ or $x \equiv 1 \bmod s$, respectively. Therefore, when s is a prime, there are no non trivial square roots of 1 modulo s.

   (b) Solution by verification, left as **exercise**. Since both 561 and 1729 are Carmichael numbers, $a^{s-1} \equiv 1 \bmod s$ for any a. You have to pick 4 different values for a and verify that for atleast 3 of them, you end up discovering non trivial square roots of 1 using the procedure explained in the question.

   (c) The following algorithm will work.

```
Primality (s)
Represent s-1 = 2^t*u where u is odd
Repeat k times {
     Pick a randomly from {1,2,3... s-1}
     If a^u = 1 mod s then return PRIME;
     Else {
             For i = 0 to t-1 {
             If a^(u.2^i) = -1 then return PRIME;
             }
          }
     Return COMPOSITE if PRIME has not been returned;
}
```

   We accept s as a prime number only if **all** the random rounds accept. If at least one round says composite, we will reject.

   Note that when s is not a prime or Carmichael number $a^{s-1} \neq 1 \bmod s$ with high probability. So none of the a^{u2^i} will be equivalent to $-1 \bmod s$. Else in the next squaring, we will get $a^{u2^{i+1}} \equiv 1 \bmod s$ which is not true for s Composite and non Carmichael.

   When s is Carmichael number $a^{s-1} \equiv 1 \bmod s$, but we have shown that with high probability, we will encounter a non trivial square

root of 1. Then on squaring, we will get $a^{u2^{i+1}} \equiv 1 \bmod s$ and then onwards 1 will remain invariant on squaring. So we will never encounter -1. So the algorithm will reject with probability.

In both cases, when s is composite, we can boost the probability by repetition and make it as close to 1 as we wish to.