Your assignment is to encrypt the following message ***"The Queen Can't Roll When Sand is in the Jar"*** using the values for p, q and e below.

From the Wiki: http://en.wikipedia.org/wiki/RSA\_(algorithm)

Key generation

RSA involves a **public key** and a [**private key**](http://en.wikipedia.org/wiki/Private_key)**.** The public key can be known to everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted using the private key. The keys for the RSA algorithm are generated the following way:

1. Choose two distinct [prime numbers](http://en.wikipedia.org/wiki/Prime_number) *p* and *q*.
   * For security purposes, the integers *p* and *q* should be chosen at random, and should be of similar bit-length. Prime integers can be efficiently found using a [primality test](http://en.wikipedia.org/wiki/Primality_test" \o "Primality test" \t "_blank).
2. Compute *n* = *pq*.
   * *n* is used as the [modulus](http://en.wikipedia.org/wiki/Modular_arithmetic) for both the public and private keys
3. Compute φ(*n*) = (*p* – 1)(*q* – 1), where φ is [Euler's totient function](http://en.wikipedia.org/wiki/Euler%27s_totient_function).
4. Choose an integer *e* such that 1 < *e* < φ(*n*) and [greatest common divisor](http://en.wikipedia.org/wiki/Greatest_common_divisor) of (*e*, φ(*n*)) = 1; i.e., *e* and φ(*n*) are [coprime](http://en.wikipedia.org/wiki/Coprime" \o "Coprime" \t "_blank).
   * *e* is released as the public key exponent.
   * *e* having a short [bit-length](http://en.wikipedia.org/wiki/Bit-length) and small [Hamming weight](http://en.wikipedia.org/wiki/Hamming_weight) results in more efficient encryption - most commonly 0x10001 = 65,537. However, small values of *e* (such as 3) have been shown to be less secure in some settings.[[4]](http://en.wikipedia.org/wiki/RSA_(algorithm)#cite_note-Boneh-3)
5. Determine *d* as:

d \equiv e^{-1} \pmod{\varphi(n)}

i.e., *d* is the [multiplicative inverse](http://en.wikipedia.org/wiki/Modular_multiplicative_inverse) of *e* mod φ(*n*).

* This is more clearly stated as solve for d given (de) mod φ(*n*) = 1
* This is often computed using the [extended Euclidean algorithm](http://en.wikipedia.org/wiki/Extended_Euclidean_algorithm).
* *d* is kept as the private key exponent.

so, d\*e= 1 mod φ(*n*) The **public key** consists of the modulus *n* and the public (or encryption) exponent *e*. The **private key** consists of the modulus *n* and the private (or decryption) exponent *d* which must be kept secret. (*p*, *q*, and φ(*n*) must also be kept secret because they can be used to calculate *d*.)

A working example

Here is an example of RSA encryption and decryption. The parameters used here are artificially small, but one can also [use OpenSSL to generate and examine a real keypair](http://en.wikibooks.org/wiki/Transwiki:Generate_a_keypair_using_OpenSSL).

1. Choose two distinct prime numbers, such as

p = 61 and q = 53.

1. Compute n = p q giving

n = 61 \times 53 = 3,233.

1. Compute the [totient](http://en.wikipedia.org/wiki/Totient" \o "Totient" \t "_blank) of the product as \phi(n) = (p-1)(q-1) giving

\phi(3233) = (61 - 1)(53 - 1) = 3,120.

1. Choose any number 1 < e < 3,120 that is [coprime](http://en.wikipedia.org/wiki/Coprime" \o "Coprime" \t "_blank) to 3,120. Choosing a prime number for e leaves us only to check that e is not a divisor of 3120.

Let e = 17.

1. Compute d, the [modular multiplicative inverse](http://en.wikipedia.org/wiki/Modular_multiplicative_inverse) of e\text{ (mod }\phi(n)\text{)} yielding

d = 2,753.

The **public key** is (n = 3,233, e = 17). For a padded [plaintext](http://en.wikipedia.org/wiki/Plaintext) message m, the encryption function is m^{17}\text{ (mod }3,233\text{)}.

The **private key** is (n = 3,233, d = 2,753). For an encrypted [ciphertext](http://en.wikipedia.org/wiki/Ciphertext" \o "Ciphertext" \t "_blank) c, the decryption function is c^{2,753}\text{ (mod }3,233\text{)}.

For instance, in order to encrypt m = 65, we calculate

c = 65^{17}\text{ (mod }3,233\text{)} = 2,790.

To decrypt c = 2,790, we calculate

m = 2,790^{2,753}\text{ (mod }3,233\text{)} = 65.

Both of these calculations can be computed efficiently using the [square-and-multiply algorithm](http://en.wikipedia.org/wiki/Square-and-multiply_algorithm) for [modular exponentiation](http://en.wikipedia.org/wiki/Modular_exponentiation). In real life situations the primes selected would be much larger; in our example it would be relatively trivial to factor n, 3,233, obtained from the freely available public key back to the primes p and q. Given e, also from the public key, we could then compute d and so acquire the private key.