Programming in Java

In the Discrete Mathematics course you learned that RSA encryption involved having a public key \((N, e)\) where \(N\) is the product of two secret primes \(P\) and \(Q\) and \(e\) is an exponent. To encrypt a message that is represented by a number \(m\) you just compute \(m^e \mod N\).

The Java `BigInteger` class contains (among others) methods called `add`, `subtract`, `multiply`, `divide` and `remainder`.

The class `String` has a method `charAt` that allows you to extract a character at a given position, and `length` to tell you how long the string is. Casting a character to an integer yields its character code.

Supposing you are given a `BigInteger` that represents \(N\) and an integer for \(e\), and not using any built-in Java methods for raising numbers to powers, write code that

(a) takes a string and encodes it as an integer; if the string contains characters \(c_0, c_1 \ldots\) the integer required will be \(c_0 + Kc_1 + K^2c_2 + \cdots\) with the constant \(K\) set to \(2^{16}\) so that the full Unicode character set can be accommodated;  

(b) encodes this number (assuming it is less than \(N\)) using the RSA method;  

(c) creates an encoded string by viewing the integer as if it was written \(d_0 + Ld_1 + L^2d_2 + \cdots\) with \(L = 26\) and then representing each \(d_1\) as a lower-case letter so that the 26 possible values are all accounted for.