Assignment: Use Maple to implement the Blum-Blum-Shub pseudo-random bit generator with 50 digit primes. Generate a string of at least 20 random bits. Print the Maple worksheet showing your work, and write the string of bits generated.

Outline and Commands:

1. First you’ll choose random 50 digit primes congruent to 3 mod 4. Enter the commands:
   
   ```maple
   > randomize();
   > nextprime( rand( 10^49 .. 10^50 )() );
   > % mod 4;
   
   When you find one that is congruent to 3, use copy and paste to name it \( p \) or \( q \).
   ```

   ```maple
   > p := paste the number here;
   ```

2. Set \( n = pq \), and choose another random number for \( x[0] \). Choose \( x[0] \) to be 50 digits or more, and relatively prime to both \( p \) and \( q \). Two numbers are relatively prime if the gcd is one. Enter the commands:

   ```maple
   > a:=rand( 10^49 .. 10^50 )();
   > gcd(a, p);
   > gcd(a, q);
   
   Both gcd’s will likely be one. If so, copy that random number to \( x[0] \). If not repeat until the gcd’s are both one.
   ```

   ```maple
   > n := pq;
   > x[0] := paste another random number here;
   ```

3. Run the following loop to generate some numbers mod \( n \). (Use Shift-Enter to go to the next line without executing the command. You can change the 4 to however many numbers you need to generate.)

   ```maple
   > for i from 1 to 4 do
   >     x[i] := x[i-1]^2 mod n;
   >     convert( x[i], binary);
   > od;
   ```

4. We can use the last \( k \) binary digits of each of these, as long as \( k \leq \log_2(\log_2(n)) \).

   ```maple
   > evalf( log[2]( log[2](n) ) );
   
   (The evalf command gives a floating point decimal approximation.)
   ```

5. Print the worksheet and write down the last \( k \) digits of each \( x[i] \). This is your pseudo random bit string.

6. Use this pseudo-random bit string to encrypt the message

   ```maple
   1111111111000000000000000000
   ```

   (That’s ten ones followed by ten zeros.)