Chapter 31 Programming

```java
public static void main(String[]args) {
    BigInteger a = new BigInteger("123");
    BigInteger b = new BigInteger("341");
    BigInteger c = new BigInteger("5");

    System.out.println("Euclid: "+euclid(b, c));
    BigInteger[] d = extendedEuclid(b, c);
    System.out.println("Extended Euclid:");
    for (int i = 0; i < d.length; i++) {
        System.out.println(d[i]);
    }
    System.out.println("Modular Exponentiation: "+modularExponentiation(a, b, c));
    System.out.println("Pseudo Prime: "+pseudoPrime(b));

}
```

```
<terminated> EuclidTesting [Java Application] C:\Us
Euclid: 1
Extended Euclid:
1
1
-68
Modular Exponentiation: 4
Pseudo Prime: composite
```

```java
public static BigInteger euclid (BigInteger a, BigInteger b) {
    if (b.equals(BigInteger.ZERO)) {
        return a;
    }
    return euclid(b, a.mod(b));
}

public static BigInteger[] extendedEuclid(BigInteger a, BigInteger b) {
    BigInteger[] result = new BigInteger[3];
    if (b.equals(BigInteger.ZERO)) {
        result[0] = a;
        result[1] = BigInteger.ONE;
        result[2] = BigInteger.ZERO;
        return result;
    }
```

For these, I simply converted the pseudocode in the book. To be entirely honest, I don't quite understand how any of the algorithms work.

```java
public static BigInteger modularExponentiation(BigInteger a, BigInteger b, BigInteger n) {
    BigInteger d = BigInteger.ONE;
    boolean [] bBits = new boolean[b.bitLength()];
    for (int j = 0; j < b.bitLength(); j++) {
        bBits[j] = b.testBit(j);
    }
    for (int i = bBits.length-1; i > 0; i--) {
        d = (d.multiply(d)).mod(n);
        if (bBits[i] == true) {
            d = (d.multiply(a)).mod(n);
        }
    }
    return d;
}

public static String pseudoPrime (BigInteger n) {
    if (modularExponentiation(BigInteger.TWO, n.subtract(BigInteger.ONE), n).mod(n) != BigInteger.ONE) {
        return "composite";
    }
    return "prime";
}
```