471-Cryptography Homework#1

Ahmet Eroğlu

210201010

In this homework I try to write 2 different functions for extended Euclidean algorithm.

```java
    public BigInteger extendedEuclideanAlgortihm(BigInteger p, BigInteger q) {

        BigInteger s = new BigInteger("0");
        BigInteger s_old = new BigInteger("1");
        BigInteger t = new BigInteger("1");
        BigInteger t_old = new BigInteger("0");
        BigInteger r = q;
        BigInteger r_old = p;
        BigInteger quotient;
        BigInteger tmp;

        while (r.compareTo(BigInteger.valueOf(0)) != 0) {
            quotient = r_old.divide(r);

            tmp = r;
            r = r_old.subtract(quotient.multiply(r));
            r_old = tmp;

            tmp = s;
            s = s_old.subtract(quotient.multiply(s));
            s_old = tmp;

            tmp = t;
            t = t_old.subtract(quotient.multiply(t));
            t_old = tmp;
        }

        x = s_old;
        y = t_old;
        return x;
    }
```

In this function I try to compute p.x+q.y=Gcd(x,y). To calculate that I take to big integer values p, q. Then initialize s, s_old, t, t_old, r, r_old, quotient, tmp. S_old is our x, t_old is our y, r_old is our Gcd(x,y).And finally return x value for using calculation Chinese remainder theorem. I write the same function again but this time I return r_old which is our gcd.

```
2
Currently Running Extended Euclidean Algorithm Finder.Enter two numbers to find EEA pairs
60 22
Result of Extended Euclidean Algortihm: -4 and 11
```

If we want to calculate x, y pairs of 60, 22, the program will calculate it with using runExtendedEuclideanAlgorithm.

```
3
Currently Running Gcd Finder.Enter two numbers to find GCD
60 22
Result of Gcd Calculation with using Extended Euclidean Algorithm: 2
```

If we want to calculate gcd of 60, 22 then we should call the runExtendedEuclideanAlgorithmGcd.

```java
public BigInteger chinese_remainder_theorem(ArrayList<BigInteger> A, ArrayList<BigInteger> Q, int k) {

    ExtendedEuclideanAlgorithm ex = new ExtendedEuclideanAlgorithm();
    BigInteger p, tmp;
    BigInteger prod = new BigInteger("1");
    BigInteger sum = new BigInteger("0");

    for (int i = 0; i < k; i++)
        prod = prod.multiply(Q.get(i));

    for (int i = 0; i < k; i++) {
        p = prod.divide(Q.get(i));
        tmp = ex.extendedEuclideanAlgortihm(p, Q.get(i));
        sum = sum.add(A.get(i).multiply(tmp).multiply(p));
    }

    return sum.mod(prod);
}
```

In the implementation of Chinese remainder theorem, the Arraylist A represent the list of modulo, Arraylist Q represent the values of that modulos. First I call the extendedEuclideanAlgorithm method and calculate x value where x is p.x+q.y=gcd(x,y) . Then calculate x = ((m.u)*b+ (n.v)*a) mod (m*n) where m, n are represented by a.get(i), u and v represents the results of extended Euclidean algorithm of m and n, a and b are represented by p value. Finally the function returns sum mod(m*n).

```
Menu:
1 : Chinese Remainder
2 : Extended Euclidean
3 : Gcd
4 : Prime Testing
5 : Exit
1
Enter variable with the given form:
First How many different modulo you will enter:
Then the modulos Finally the remainders for that modolos
For example if we want o enter 2 modulos And their values will be 3 mod5 and 2 mod3
Your input should be: 2 3 5 2 3
2 12345 11111 7 3
Result of Chinese Remainder Theorem: 109821127
```

The final part of the implementation is primality testing;

```java
    public static boolean checkPrime(BigInteger n, int maxIterations) {
        if (n.equals(BigInteger.ONE))
            return false;
        for (int i = 0; i < maxIterations; i++) {
            BigInteger a = getRandom(n);
            a = a.modPow(n.subtract(BigInteger.ONE), n);

            if (!a.equals(BigInteger.ONE))
                return false;
        }
        return true;
    }
```

This function checks a number that is prime or not, to do that is generates a random number and try to calculate “(a^ (n-1)) mod n” where n is our number that is currently testing and a is a random variable. The function try this steps maxIteration times because there is some numbers that called strong liar that can says its prime with that value but actualy number is composite to avoid that program try maxIteration times then says it is prime or not.

```
Menu:
1 : Chinese Remainder
2 : Extended Euclidean
3 : Gcd
4 : Prime Testing
5 : Exit
4
Currently Running Prime Testing.Enter a number to test it.
67280421310721
probably prime
4
Currently Running Prime Testing.Enter a number to test it.
67280421310723
non prime, composite
```

I try the program with a big prime integer 67,280,421,310,721 the program call it prime.