CS211 ALGORITHMS & DATA STRUCTURES II

LAB 6

Dr. Phil Maguire

PUBLIC KEY CRYPTOGRAPHY

Pen and Paper Exercise

Your public key is (29, 2, 3) and your private key is 5. Bob uses your public key to send you a message. He sends you the cipher (23, 27). Use your private key to obtain the secret message code number. Read over the lecture notes to get the required formulas.

Programming Exercise

NOTE: THIS LAB IS GOING TO BE HACKERRANKED

THE EXAMPLE BELOW IS A SINGLE TEST CASE ONLY

Alice's public key is (24852977, 2744, 8414508). You eavesdrop on the line and observe Bob send her the cipher (15268076, 743675). Extract the message by any means.

Warning: for the programming part, make sure to use *longs* rather than *ints* (you may need to put an 'l' at the end of the number). Also, make sure to keep modulo-ing every time the number in the calculation gets a little too big – never do large power multiplications straight off because Java cannot process large numbers like this.

As a result, it might be worthwhile to use these recursive methods for modulo multiplication and modulo raising to a power. These work efficiently and ensure that the numbers never get too big.

```
    public static long modPow(long number, long power, long modulus){
//raises a number to a power with the given modulus
```

```
//when raising a number to a power, the number quickly becomes too large to
handle
//you need to multiply numbers in such a way that the result is consistently
moduloed to keep it in the range
//however you want the algorithm to work quickly - having a multiplication
loop would result in an O(n) algorithm!
//the trick is to use recursion - keep breaking the problem down into smaller
pieces and use the modMult method to join them back together
        if(power==0)
            return 1;
        else if (power%2==0) {
            long halfpower=modPow(number, power/2, modulus);
            return modMult(halfpower,halfpower,modulus);
        }else{
            long halfpower=modPow(number, power/2, modulus);
            long firstbit = modMult(halfpower,halfpower,modulus);
            return modMult(firstbit,number,modulus);
        }
    }


     public static long modMult(long first, long second, long modulus){
//multiplies the first number by the second number with the given modulus
//a long can have a maximum of 19 digits. Therefore, if you're multiplying
two ten digits numbers the usual way, things will go wrong
//you need to multiply numbers in such a way that the result is consistently
moduloed to keep it in the range
//however you want the algorithm to work quickly - having an addition loop
would result in an O(n) algorithm!
//the trick is to use recursion - keep breaking down the multiplication into
smaller pieces and mod each of the pieces individually
        if(second==0)
            return 0;
        else if (second%2==0) {
            long half=modMult(first, second/2, modulus);
            return (half+half)%modulus;
        }else{
            long half=modMult(first, second/2, modulus);
            return (half+half+first)%modulus;
        }
     }
```

Bonus Question (3 Maynooth Coins for first 3 people to email me)

A 28-minute audio file called secret.m4a has been uploaded to Moodle. This file was generated using the Morse Code program you can see on Lab5Solution. A well-known English text file was simply fed to this program – but what text file? As you can see, spaces have been taken out, reducing it to binary, but using a scrambled International Morse Code protocol. What is this famous piece of text? The first three people to email me the answer will be rewarded.