Fall 2022 CIS 3362 Homework #6: Public Key Encryption
Check WebCourses for the due date

1) (10 pts) In the Diffie-Hellman Key Exchange, let the public keys be p = 61, g = 31, and the secret keys be a = 45 and b = 16, where a is Alice's secret key and b is Bob's secret key. What value does Alice send Bob? What value does Bob send Alice? What is the secret key they share? Use a program or calculator to quickly simplify the modular exponentiations that arise, but show what each calculation is.

2) (10 pts) In an RSA scheme, p = 43, q = 23 and e = 181. What is d? Show the work by hand, but for any complicated calculation, do it on a calculator or use a program. (So, show each step of the Extended Euclidean Algorithm, but feel free to use a calculator to quickly get quotients and remainders.)

3) (20 pts) The Fast Modular Exponentiation code shown in class is written in a top-down recursive fashion. The problem can be solved with similar efficiency iteratively, by simulating iterative base-conversion (to binary) formula with the exponent, and calculating the base raised to each successive power of 2 (mod the mod value). Here is a trace by hand of this algorithm with base = 3, exponent = 43, mod = 83:

Store our initial result as 1. We'll multiply terms of the form 3^x mod 83 into this result.

43%2 = 1, this means that 3^1 is a part of our result, so change our result to 3.
43/2 = 21 (remaining part of our exponent.
Update our current multiplying value from 3 to $3^2 = 9$.
21%2 = 1, this means that 3^2 (already stored) is part of our result so change result = 3 x 9 = 27
21/2 = 10
Update our current multiplying value from 9 to $9^2 = 81$ (mod 83). [Note: this is 3^4.]
10%2 = 0, so we don't change our result
10/2 = 5
Update our current multiplying value from 81 to 81^2 (mod 83) = 4 [Note: this is 3^8 mod 83.]
5%2 = 1, this means that we update our result = 27 x 4 (mod 83) = 25.
5/2 = 2
Update our current multiplying value from 4 to $4^2 = 16$. [Note: this is 3^{16} mod 83.]
2%2 == 0, so we don't change our result
2/2 = 1
Update our current multiplying value from 16 to 16^2 (mod 83) = 7. [Note: this is 3^{32} mod 83.]
1%2 = 1, this means that we update our result = 25 x 7 (mod 83) = 9
1/2 = 0, so we exit our loop

The answer is 9. Notice, that what's really happening here is that $43 = 101011_2$, so we have $3^{43} = 3^{32}3^8 3^2 3^1$. The algorithm has one value that starts at 3, and then gets squared each time (generating 3^2,3^4,3^8, etc.) As we start with our exponent at 43, each time we mod and divide, we're peeling off the last binary digit. If this is one, then we can just multiply our current term (power of 3) and mod into our accumulator variable storing the result.

Please write this function in either Python or Java (so we can easily test its efficiency on large integers!). Please use one of the following two prototypes:

```
# Python
def fastmodexp(base,exp,mod)

// Java - Please put in class Question3
public static BigInteger fastmodexpo(BigInteger base, BigInteger
exp, BigInteger mod)
```

Our TAs will grade your function by making calls to it from their own main.

4) (30 pts) The following ciphertext below was created with the El Gamal cryptosystem with the prime q = 19157. You also know that the plaintext was written in all lowercase letters and split into blocks of 3 characters and the way that the characters abc were mapped to a number was using the formula 676*value(a) + 26*value(b) + value(c), where the value function returns the 0 to 25 value typically used in cryptography to represent letters ‘a’ (0) to ‘z’ (25). Use this information to decrypt the following ciphertext:. (Note: This will also be given to you in a text file, for ease of processing and as mentioned, each line represents the encryption of 3 characters, the first number is c1 and the second number is c2.)

```
8352 17131
1565 17613
18077 12139
14140 1640
8910 13256
18846 1135
6812 12961
8620 4761
3984 3064
3129 6296
16594 4682
17932 9494
7407 14354
1869 1038
10542 7222
7492 12721
13572 1255
16299 11712
7197 16361
17245 6683
6136 6014
18340 6702
16036 631
18012 12760
6114 7446
9276 10932
11176 18217
9278 2044
11461 15736
10544 7635
12477 5558
17923 5156
16499 19045
12370 5390
2004 2128
```

```
17590 7949
513 13666
3127 14850
13686 3243
12928 12825
18173 18480
5764 8808
5244 12211
2234 9429
18401 17242
9050 11469
13130 11932
12471 11967
571 3303
16719 16142
1727 19137
17036 3174
344 12895
4124 4096
5997 2360
925 3465
1078 5128
13291 10623
2626 615
```

5) (30 pts) Time to break another code! This was produced using rsa3.py. Here are the public keys for the system used.

```
Public key n = 27299026172500142606639
Public key e = 7455573413522025488689
```

Here is the ciphertext to decipher:

```
17522509615978177173541
9355128999191425516583
17969916466888149264201
16284171845378592258094
9091719398917520846644
1858849677057002482192
10113610280720608183543
15541430701657029579837
3153597330798750301099
11222381486914003429676
17779392253507960098764
19095139734721053624828
12524940225900268861892
```

Each number represents a block of 16 **lowercase letters.**

Good luck!

Arup