ECE 3020 Homework 6
Due Date: Friday, October 13, 3:00 PM

Large prime numbers are used in the generation of cryptographic keys for encryption, decryption, and digital signatures. (See http://www.di-mgt.com.au/rsa_alg.html for a description of the complete key generation algorithm for the RSA cryptosystem.) In this assignment, you will write code to implement a randomized algorithm to generate very large prime numbers and you will use a tool to generate encryption keys based on prime numbers that you supply.

Part 1 – Large Prime Generation (8 points)

In this part, you will use the randomized prime-testing algorithm described on page 219 in Chapter 4 of the on-line text to generate large numbers that are prime with a very high probability. Set k=10 for the algorithm. Generate random numbers of at least 20 digits until 2 numbers pass the prime test. For each number tested, output the remainder value for each of the 10 trials and the result (prime or not prime). You can test the correctness of your code on shorter (but still large) prime numbers from an on-line list, e.g.

http://www.primos.mat.br/indexen.html

Note: For this part, you will have to use a large integer class. In Java, there is a BigInteger class that is part of the standard math package. Python has native support for large integers. In C++, one example of a large integer class can be found at https://mattmccutchen.net/bigint/. For C, there is a large integer library available at http://gmplib.org/

Also, for the program to run efficiently and work on very large integers, you will need to perform modular exponentiation. Functions for modular exponentiation are typically built into large integer classes (all of the above-mentioned classes/libraries contain such a function). You ***must*** implement the prime-testing algorithm in your code, i.e. you are ***not*** allowed to use built-in functions for prime testing that might come with these libraries.

Part 2 – Cryptographic Key Generation (2 points)

In this part, you will use the on-line RSA calculator at:

https://www.cs.drexel.edu/~introcs/Fa11/notes/10.1_Cryptography/RSAWorksheetv4d.html

to generate encryption keys and use them to encrypt and decrypt short messages. Unfortunately, this calculator does not work with very large prime numbers. Modify your code from Part 1 to generate two prime numbers that are each at most 12 bits long. Enter these primes into the RSA calculator and follow the instructions to generate the modulus n, the public encryption key e, and the private encryption key d. *Note:* You will have to keep trying different candidate K values and factors of K until you find an e, d pair for which e*d = 1 mod r. Once you find an appropriate e, d pair, you can use them to encrypt and decrypt messages in Step 4 on the calculator.

Turn-in:

For Part 1, turn in your source code and a screen shot of the output of your program displaying two numbers of at least 20 digits that are very likely prime. For Part 2, turn in screen shots showing:

- your program output showing the two primes used for this part
- all steps of the RSA calculator from entering the primes to encrypting and decrypting a message