ICS440/SEC540

Cryptography and Blockchain Applications

TERM 201 – Sultan Almuhammadi

Programming Assignment – PA1

Due: Saturday, Oct 17

In this assignment, you will implement some algorithms discussed in the class. You may use these algorithms as building blocks later in your project. Any programming language can be used as long as it allows large integers (about 300-600 digits). Many programming languages have special libraries and classes for large integers (for example, Class BigInteger in Java). Python is highly recommended since it allows large integers by default. You may self-learn Python (in 2 hours) by watching a number of short tutorials (see Course Materials). Notice that help in PA's and the project is only provided in Python. If you choose any other language, make sure your programming skill in that language is adequate.

Problem Set: [100 points]

Q1. [10 pts] Install your favorite programming language compiler and IDE. Familiarize yourself with the environment. (Hint: For Python, watch Tutorials #1 and #2, follow the steps to install Python and Jupyter Notebook through Anaconda, and learn about variables and If-statements in Python).

Q2. [10 pts] Learn about functions, arrays and loops. Practice by implementing small programs and functions to test your understanding. (Hint: Python users watch Tutorials #3 to #7). Create an array *my_Primes* that contains all the primes less than 20. Make a loop that prints these elements one by one. Save this code for submission.

Q3. [30 pts] Implement the *gcd* function using the Euclidean Algorithm to compute the gcd(a,b). (Hint: You may implement this function recursively).

Q4. [40 pts] Implement a primality test function *is_prime* (*n*, *k*) based on Fermat Little Theorem, where *n* is the number to be tested, and *k* is the number of bases to be used. The function should return False if *n* is not prime, and True if *n* is pseudoprime to these *k* bases. Notice that *k* is a small constant, and therefore, some composites will be counted as primes. (Hint: for small *n,* you may first check *my_Primes* array in Q2 before applying the FLT-based test).

Q5. [10 pts] In the main program, use your *is_prime* function in a loop, with a reasonable value of *k,* to print all prime numbers less than 2000. (Your code should work with large integers as well)

Submission: (Deadline: see Blackboard)

What to submit? Submit all your codes and the main program. You can either keep them in one file or separated files saved as: q2, q3, q4 and q5 (with appropriate extension for the programming language), then zip them in one file.
File Name: Save the zip file as **PA1.csn##.zip**, where ## is your CSN. (For Python users, you may submit one Jupyter-notebook including all the codes, saved as **PA1.csn##.ipynb**)