CS315, Fall 2012
Homework 3 (bonus worth 100)

due: Oct 8

Problem 1: Implement an algorithm to generate prime numbers. You will need to implement the following ingredients (some of them you developed for earlier assignments):

1. A method to generate random binary numbers with n-digits (hint: for the most significant digit, you have no choice, it will be 1; for all other position, generate 0 or 1 at random)
2. A method to compute modular exponentiation (implement the algorithm discussed in class, it will require that you also implement the algorithms for division by 2, and for modular addition and multiplication).
3. A method to test primality. Since the numbers you will be testing are randomly generated, use the first algorithm we discussed with a single test $3^{N-1} \equiv 1 (mod\ N)$
4. A method to convert from binary to decimal

Test your algorithm on randomly generated 16-bit numbers. That is, use your algorithm to generate 20 numbers (presumably prime), and check how many of them are primes by a brute-force approach of trying all divisors up to the square root of the number; you will be able to use regular integer arithmetic to do so). How many times did your algorithm come back with a wrong answer?

Use your algorithm to generate prime numbers with 64, 128, 256 and 512 binary digits. In each case, report the number of randomly generated numbers until the prime was found. What should it be in theory? Are your results consistent with the theory?

In your write up, report the 256 and 512-bit prime numbers you generated, given in decimal.

Report CPU times taken by your algorithm to generate primes with 64, 128, 256 and 512 binary digits. Based on the times, estimate the rate of growth. Is it consistent with the theoretical worst-case behavior of the algorithm to generate primes we derived in class. Provide explanations.

Carefully document your code, and describe your implementation.

Important: 1. As in other assignments, you need to use one-dimensional vectors to represent long binary numbers. Please, get in touch with me if you are still uncertain how to do it.
2. Do not delay starting to work on that assignment. It will take you some time. Moreover, even if your algorithms are implemented correctly, you may be frustrated by the slow running time. “Chunking” several (30) bits into a single vector entry will make your programs work much faster.

Solutions (other than program code, and associated readme files) must be typed (wordprocessed), and submitted by e-mail as a pdf document