CS 4390/5390 Fall 2013
Shirley Moore, Instructor
Final Project

The final project will consist of:

1. implementation and/or use of a significant number theoretic algorithm or set of algorithms not included in a homework assignment
2. a short paper describing the mathematical background for the algorithm(s) and your design choices
3. a presentation during the final exam period describing and demonstrating your program and the underlying theory

The specific algorithm(s) can be of your choosing but you must have your topic pre-approved by the instructor. You may work individually on the final project or in teams of up to three people. In the case of group work, you must clearly document the contributions of each team member and carry out the amount and difficulty of work proportional to the size of your team. For example, a team project might implement and compare a set of related algorithms, or compare various parallelization strategies for a given algorithm (if you know parallel programming).

Suggested topics (pre-approved):

A. Primality testing and certification

Compare several different primality testing algorithms in terms of runtime and effectiveness. Also implement, or select and learn how to use, a primality certification algorithm that will show without a doubt that a number is prime. You do not need to implement the algorithms yourself and you may make use of the resources at http://primes.utm.edu/prove/prove4.html as long as you explain and justify your choices.

B. RSA implementation

Implement a secure RSA cryptography system that includes tools for generating a public-private key pair and for encrypting and decrypting messages. The tool for generating the public-private key pair should allow the user to specify the number of bits in the modulus and optionally the public exponent e. Be sure to use appropriate armoring as described at http://rdist.root.org/2009/10/06/why-rsa-encryption-padding-is-critical/ .

Normally RSA encryption is not used by itself to send messages. Rather, the content to be sent is first encrypted under a content-encryption key using a symmetric encryption algorithm, and then the content-encryption key is encrypted using the RSA public keys of the recipient of the content. The encrypted content-encryption key and the encrypted content are then represented together in the form of a digital envelope. If you are doing a group project, you may choose to implement such a hybrid message encryption scheme.

You may make use of a library of routines such as those at https://www.openssl.org/docs/crypto/rsa.html but you must implement your own tools.

C. Factoring challenge.

Implement a scheme that will attempt to factor an arbitrary large integer. Show that your scheme works well for various types of composite integers and solves easy cases quickly. Practical tips:

1. Don't try to factor a prime. Always perform at least a probable prime test before starting the factoring process.
2. Look for small factors first.
3. Use the methods appropriately. Try one or more category 1 methods first. Don't use the quadratic or number field sieve algorithms until simpler, faster algorithms have been tried. Choose the parameters for all factoring algorithms appropriately.
4. Remember that some factoring algorithms are probabilistic. Do not expect prime factors to be discovered in order of size.
5. When factoring an integer thought to have no small factors, choose the best algorithm for the size of the number. The quadratic sieve is fastest for 50- to 100-digit integers, and the number field sieve is fastest for numbers greater than 100 digits.
6. When you find a factor, test it and its codivisor for primality.

You may make use of existing factorization libraries such as TIFA at http://www.lix.polytechnique.fr/~milanj/tifa/tifa.xhtml and those described at http://www.boo.net/~jasonp/qs.html .

D. Prime counting function

Implement an efficient method to compute the prime counting function.