Prof. Sergio A. Alvarez
Fulton Hall 410–B
Computer Science Department
Boston College
Chestnut Hill, MA 02467 USA

http://www.cs.bc.edu/∼alvarez/
alvarez@cs.bc.edu
voice: (617) 552-4333
fax: (617) 552-6790

CS383, Algorithms
Spring 2009
HW3

1. Consider the following algorithm for computing modular remainders:

   Algorithm 1: recMod
   Input: Two integers $a \geq 0,\ b \geq 1$.
   Output: The modular reminder $a \pmod b$.
   recMod$(a,\ b)$
   if $a = 0$ **then return** 0
   rem $= 2$ recMod$(\lfloor a/2 \rfloor,\ b) + a \pmod 2$
   if rem $< b$ **then return** rem
   else return rem $- b$

   (a) Trace the execution of Algorithm 1 on input $(15, 4)$ by drawing the recursion tree for recMod$(15, 4)$. Include the value returned by each invocation (node).

   (b) Give a general argument to show that recMod(a, b) correctly computes the modular remainder $a \pmod b$ for any integers $a \geq 0$ and $b \geq 1$.

   (c) Notice that each of the arithmetic operations in the recursive call in Algorithm 1 involves the number 2 as one of the operands. On computers that use binary arithmetic at the machine level, these operations can be performed quickly: multiplication and integer division by 2 reduce to left and right shifts by one bit, and the remainder modulo 2 just involves testing the least significant bit. Therefore, we will assume that these three operations may be performed in time $O(d)$. Addition and subtraction of d-digit numbers are also assumed to take time $O(d)$. Analyze the asymptotic running time of Algorithm 1 on d-digit inputs a and b, keeping these comments in mind. Explain in detail.

2. Consider the following "beefed up" RSA encryption scheme (my apologies to any vegetarians). The recipient picks *three* large primes p, q, r and a number e between 1 and $(p-1)(q-1)(r-1)$ that is relatively prime to $(p-1)(q-1)(r-1)$, and publishes the pair $(N = pqr,\ e)$ as his public key. The encryption of a message m (number between 0 and $N-1$) is the quantity

   $$\text{encrypt}(m) = m^e \pmod N$$

   (a) Show that the above encryption function is invertible by explicitly computing its inverse function. Proceed by analogy with the discussion of RSA from class and the textbook. Include a step-by-step justification of your answer. You'll need Fermat's little theorem.

(b) Is the proposed scheme cryptographically secure? That is, if someone were to intercept the transmitted encrypted message encrypt(m), would it still be very difficult for them to recover the original message m based only on encrypt(m) and the public key (N, e)? Discuss, paying particular attention to the time complexity of computing the inverse function in the preceding subtask.

3. Solve the last task in HW2. I suggest that you program suitable functions to implement integer factoring, the extended Euclidean gcd algorithm, and modular exponentiation.