Homework 2

This homework is due on *Monday, February 3, 2014* by 10:39:00AM. There are **25 possible points**. You must submit your homework on paper in lecture. We do not accept emailed or otherwise electronic submissions (zero points). Late submissions or submissions that do not follow the strict policies from the syllabus will receive a zero. Please contact the TA well ahead of the deadline if you have a question about these procedures.

Submit each homework[1] problem stapled separately. We will have separate boxes at the front of the lecture hall for submitting each problem. Points will be deducted for solutions that do not follow this process. On the front page of **each** stapled problem, the top right corner must provide the following items:

- Print your name
- Print your uniqname
- Print the homework-problem number.

Example:

Name: John P. Doe
Uniqname: johnpdoe
Problem: 2-1

If your solution spans multiple pages for a problem, mark the top of each page with your initials. Some problems contain multiple components. For each component of a problem, write a descriptor of the component. Points may be deducted if your TA has problems understanding or reading your solution. In mathematical problems, **show all your work**. If you receive any key insights from someone else or some other resource, you must cite that person or resource.

Problem 2-1. Fear Factor (5 pts)

Many cryptosystems will fail if factorization in general of composites becomes computationally easy/efficient (i.e., polynomial time). In cryptographic systems, the defender of a system will often find it convenient to generate a composite by first generating a set of primes, then multiplying to produce a composite of a known factorization. In this way, we can easily compute $\phi(n)$ using one of our standard formulas. Show mathematically that if one can easily compute $\phi(n)$, then one can also easily factor n into its constituent prime factors. That is, construct a polynomial time algorithm that takes a positive integer as input and produces its prime factorization, after given an efficient subroutine that can compute $\phi(n)$.

[1] We will distribute our favorite solution for each problem to the class as the "official" solution—this is your chance to become famous!

Problem 2-2. How do I GCD thee? Let me count the ways. (5 pts)

In lecture, we provided the classic recursive definition for computing GCD:

$gcd(a, b) = gcd(b, a \bmod b)$ accompanied by its base case.

However, it turns out that computing a modular reduction is surprisingly hard for computers unless the modulus is a power of two. Another way to implement a fast GCD is to avoid modular reductions entirely and instead focus on bit shifts that represent division by two. We provide a *partial* specification of an alternative GCD algorithm:

$$gcd(a,b) = \begin{cases} \cdots & \\ gcd((a-b)/2, b) & \text{if } a, b \text{ odd} \\ gcd(a/2, b) & \text{if } a \text{ even}, b \text{ odd} \\ gcd(a, b/2) & \text{if } a \text{ odd}, b \text{ even} \end{cases}$$

Your task is to complete the specification of this version of GCD that avoids modular arithmetic and is fast on computer hardware. Implement both versions of GCD and measure their performance. For instance, one could use the "time" function from `time.h`. You are free to use other timing methods and trade suggestions on Piazza on how to measure time (and what time means). You may not share your code or algorithm. We recommend using C++ with the `libgmp` big number library. If you wish to use a different programming language, first get approval from the TAs on Piazza. We do not provide support for other languages.

The performance measurements should look at an interesting number of *bit lengths* that increase exponentially. The x-axis of the graph should be bit length. The y-axis should be performance time with an approporiate unit of measurement. For instance, look at bit lengths for the a, b arguments that range from $k = 1, 2, 4, 8, 16, ...$ until you feel you have reached an interesting asymptotic behavior for the running time. That is, iterate over k, then randomly select inputs of length k bits to give as input to each algorithm being measured. It's OK to let a, b have the same bit length for all tests. Note that the most significant bit of a k-bit number is 1 for unsigned numbers. Kudos for students who provide error bars in graphs that meaningfully indicate statistical confidence when taking multiple samples for each bit length k to factor out timing noise on the computer.

Provide your completed specification of this new GCD algorithm, print out your source code, and provide your performance graph(s). Do not upload to CTools. Readable and elegant code will receive the most points. Provide at least a few interesting test cases of input in decimal (i.e., pairs of inputs to GCD) along with the output in decimal. One of the test cases should have at least six decimal digits for each input. Explain which algorithm runs faster when and why.

Problem 2-3. Least common material for relatively bad fashion (5 pts)

Alex, Emily, and Prof. Fu invested in a large number of U-M CSE T-shirts. Alex managed to nab three shirts: blue, maize, and white. Emily managed to procure five shirts: blue, maize, white, tie-dye blue/maize, and an XKCD-themed CSE shirt. Prof. Fu found only two shirts: one blue, one maize. Logically, the staff wear their shirts in strict order. Alex wears blue, maize, white, and then repeats in that order after laundry. Emily does the same with her five shirts, and Prof. Fu does the same with his two. Assume that the staff have a seven-day work week and change shirts each day.

(a) Fashion that

How often will Alex and Emily wear the same color T-shirt during Friday discussion? Provide an answer measured in days, and explain your reasoning.

(b) In a modular fashion

It's never a dull moment in EECS 475 because it's rare for the staff to repeatedly all wear the same configuration of clothing. Assume that the four-tuple (A, E, K, Day) represents the fashion configuration of the staff Alex, Emily, and Prof. Fu on a given day. Let the T-shirts blue, maize, white, tie-dye blue/maize, and XKCD respectively map to identifiers $\{0, 1, 2, 3, 4\}$. Also let the days of the week Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday map to $\{0, 1, 2, 3, 4, 5, 6\}$ respectively. So if Alex wears maize, Emily wears XKCD, and Prof. Fu wears blue on a Wednesday then $(A, E, K, Day) = (1, 4, 0, 2)$.

How long can the staff maintain unique fashion schemes before it's possible for a student to spot the staff wearing the same clothes on the same day of the week? That is, how many days maximum can the staff ensure that (A, E, K, Day) does not repeat? Provide an answer measured in days, and explain your reasoning.

(c) OU812

In an effort to provide better educational experiences, the University has established a Blue Ribbon Committee to consider an 8-day work week. The number eight was chosen because CSE faculty demanded a power of two after hearing that peer schools were considering powers of one. The 8th day will be called Blueday, and classes will run every day. We now map the days Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday, Blueday to $\{0, 1, 2, 3, 4, 5, 6, 7\}$ respectively.

How many days maximum can the staff ensure that (A, E, K, Day) does not repeat on the same day in this new 8-day work week? Provide an answer measured in days. Explain using theoretical tools from class why this number is what it is.

Problem 2-4. Inverses and Modular Exponentiation (5 pts)

Do problem 5.3 (page 226) from Stinson. But instead of using the Extended Euclidean Algorithm, compute inverses using the Euler extension to FLT. Exponentiation should be done with square and multiply. All steps of exponentiation need to be shown. Show all your work and verify correctness by showing that the element multiplied by its inverse = 1.

Problem 2-5. Say What? (5 pts)

See Piazza. You know what I meme.