CSE596, Fall 2014 Problem Set 1 Due Fri. Sept. 12

Lectures and Reading. This week's lectures are covering formal languages (section 1.1 plus aspects of 1.5) and regular expressions (notes on webpage), then moving on to Turing Machines (sections 2.1–2.3). Sections 1.3–1.4 and 1.6 will be covered formally when the material is needed later; for now it is enough to appreciate from last week's lectures that the integers and rational numbers can both be "enumerated" the same way the natural numbers are listed out. That this *cannot* be done for the real (or complex) numbers will be proved when we hit sections 3.2 and 3.4, for which I will also give supplementary reading.

Next week's lectures will portray Deterministic and Nondeterministic Finite Automata as special cases of Turing Machines. This is also why one should read sections 2.1–2.3 in one gulp, since the idea of nondeterminism comes in 2.3.2. There are some things in those sections that won't be covered quite yet, such as the formal notation for *computations*—but note that my use of the "turnstile" symbol for saying "3 goes-to 10 goes-to 5 goes-to. . . " in the Collatz conjecture conveys the essential idea, so hopefully helped for motivation while reading it. Lectures will pick-and-choose from the notes by Mitsunori Ogihara on my course webpage, emphasizing the theorems translating from regular expressions to NFA's to DFA's and back again. The Myhill-Nerode handout may not get covered until Week 4.

Office Hours: My regular office hours will include 1–2pm on Tuesdays and Thursdays, and probably 3:15–5pm on Mondays will be the rest of the time. For reasons of schedule and Mike Wehar's necessary absence until next week, a time for a regular Review Session and our final office-hours schedule are still TBA.

Turing Kit Simulator (optional): This appears to work again, at least from the CSE Linux terminals in Davis Hall and other labs. Directions for setting it up are on the course webpage.

Assignments: For homeworks, we are experimenting with ideas between recent past policy of not grading some problems, and my usual high value placed on written homework. Here let us try having a mix of "study-guide problems," which have small (3 pt.) "checkoff credit" for giving an earnest answer, and regular graded problems. The study-guide problems (labeled A,B. . .) are open-ended in a way that hopefully lessens the perceived responsibility on your part to give a full "closed" answer (such as a proof); it is enough to try to grasp the idea being expressed. All four problems on this set are for hardcopy submission on **Friday, 9/12**, in class period. Depending on schedule the due date might move to Wednesday, and similarly, the First Prelim Exam might be Fri. Oct. 3, Wed. Oct. 8th, or Fri. Oct. 10th.

(A) What difference(s) do you perceive between the nature of trying to solve the following two tasks by an algorithm, given a number n?

(a) Testing whether $2n + 2$ is the sum of two prime numbers.

(b) Testing whether the "Collatz process"

```
if (n % 2 == 1) { n = 3*n+1; } else { n = n/2; }
```

when begun at n eventually hits 1.

How would this matter if you were given a range 1..N and had to do the test for each $n = 1$ to $n = N$ in sequence?

(B) Regard a language L as a subset of the natural numbers. Define the real number x_L by adding $2^{-(n+1)}$ if the number n belongs to L. That is, put a '1' in the $(n+1)$st decimal place, which corresponds to $1/2^{n+1}$, if and only if $n \in L$. So now we are considering $\pi - 3$ in binary, which is 0.0010010000111111, to come from $L = \{2, 5, 10, 11, 12, 13, 14, 15\}$. Also 1/3 comes from the odd numbers and 2/3 from the even numbers—noting the standard rule that *zero counts as an even number.* (Again this is shifted by 1 from before when we numbered the binary-decimal places starting from 1 not 0.)

Is going from L to x_L a *one-to-one correspondence*? Or are there two different sets that go to the same real number? What if we limit attention to *infinite* languages L? [This references an early part of section 1.6, but does not depend on the difficult material about uncountability, which is not yet covered.]

(1) Design a deterministic finite automaton (DFA) M that accepts a binary string x if and only if the number of 1's in x is a multiple of 6. What design difference would it make if you instead chose not to count zero as a multiple of 6? Then write a regular expression for the language. (9+3+6 = 18 pts.)

(2) Now design a DFA that recognizes the language L_6 of numbers that are multiple of 6 in *standard binary* (not 2-adic) encoding. The numbers are allowed to have leading 0's. Thus 0^+ is a subset of L_6, but the empty string λ itself is not considered to belong. Some other members of L_6 are $110, 1100, 10010, 0110, 010010, \ldots$ Design the DFA in a way that if we asked instead for numbers of the form $6m + a$, $1 \leq a \leq 5$, you could just change the accepting states. Do you need more states than you used in Problem (1)? (15 + 3 = 18 pts., for 36 total, 42 including checkoff credit)