CS 321
For Thursday 28 Jan

Homework #3
Turing Machines

1. Use the web to find a reasonable Turing machine simulator. Use the Turing machine simulator that you have located to simulate several Turing machine programs.

   Start with some easy problems, for example, binary addition, and
   determining if an input string is a palindrome.

   Next do at least one harder problem, for example,
   computing Fibonacci numbers (in base notation),
   or multiplying natural numbers (in base notation),
   or determining if an input number is a perfect square,
   or determining if an input number is a prime.

   (a) What Turing machine simulator did you use?

   (b) What was the first simple problem you looked at?
   If n was the length (in characters) of the input, what n's did you try?
   In terms of n about how many steps did the Turing machine execute?

   (c) What was the second simple problem you looked at?
   If n was the length (in characters) of the input, what n's did you try?
   In terms of n about how many steps did the Turing machine execute?

   (d) What was the harder problem you looked at?
   If n was the length (in characters) of the input, what n's did you try?
   In terms of n about how many steps did the Turing machine execute?

2. The **UTM** (universal Turing machine) operates with a fixed alphabet.
   We claim that the **UTM** can simulate any Turing machine, but the Turing machine (to be simulated) may have an input alphabet Σ (for example the Greek alphabet) and an output alphabet Ω (for example the English alphabet) which are different than the **UTM**'s alphabet.

   Explain how a string over Σ can be converted to a string over the **UTM**'s alphabet, a process we can call CODING.
   Explain how a string over **UTM**'s alphabet can be converted to a string over Ω, a process we can call DECODING.

   Explain how CODING and DECODING can be carried out by VERY SIMPLE Turing machines.

3. Let E be Euclidean geometry expressed as a *formal system.*
   There is a set of *axioms* (assumed basic truths), and a set of *rules* which allow one to prove new truths from some combinations of known truths.
   The *theorems* of this system are ALL the truths which can be obtained by starting with axioms and using the rules.
   Show (high-level argument) that there is a Turing machine which is an **acceptor** for the set of *theorems* of E.
   Is there a Turing machine which is a **recognizer** for the set of *theorems* of E.

4. Are TWO HEADS BETTER than ONE?
   Show that a Turing machine with TWO read-write heads can be simulated by a Turing machine with only one read-write head.
   (You may need some assumptions on how the TWO heads work together.)

5. TWO HEADS are BETTER than ONE
   Show that a a Turing machine with TWO read-write heads can compute some things faster than a Turing machine with only one read-write head.
   Specifically, describe a *recognition* problem which takes in strings of length n and outputs either YES or NO depending on whether or not the input string is in the desired set. Pick your set so that

   (a) Every one head TM takes time at least $O(\, g(n)\,)$
   (b) There is a two head TM which solves this recognition problem and uses time $O(\, f(n)\,)$
   (c) Show that $O(\, f(n)\,) \lneq O(\, g(n)\,)$
   (d) (In case you don't remember) $f(n) = O(\, g(n)\,)$ iff $\exists c > 0 \quad f(n) \leq c\, g(n)$ for all big enough n.

6. Kleene's Normal Form Theorem states:

   Any computable function
   can be computed by a program with at most one **WHILE** *loop.*

   Demonstrate the Kleene Normal Form Theorem by showing how to simulate any Turing machine by a program with one WHILE loop.
   (Hint: Use a CASE statement.)