Write every problem on a separate sheet of paper, write *legibly*, and write your name, your student ID, this Homework # and the problem # on top of each page.

The description of your proofs should be as *clear* as possible (which does not mean *long* – in fact, typically, good clear explanations are also short.)

Make sure you are familiar with the collaboration policy, and read the overview in the class homepage theory.cs.stanford.edu/∼trevisan/cs154.

Recall that our late policy is that late homework is not accepted (however the lowest homework grade is dropped).

Refer to the class home page for instructions on how to submit the homework.

Homework 3

Due on Thursday, February 2, 2012, 9:30am

1. (3 points) Recall that in the last problem set, we showed that the language

$$L = \{x@y@z | x, y, z \in \{0,1\}^* \text{and } \text{bin}(x) + \text{bin}(y) = \text{bin}(z)\}$$

   is not regular. Show that this language is decidable by defining a Turing machine that decides it. You should describe this Turing machine in some detail, but not to the point of actually specifying the states and drawing a diagram. The level of detail used in Sipser Examples 3.11 and 3.12 is ideal.

2. (4 points) Let $\prec$ be the lexicographical order on strings in $\{0,1\}^*$. In other words, $s_1 \prec s_2$ iff either $|s_1| < |s_2|$, or $|s_1| = |s_2|$ and $bin(s_1) < bin(s_2)$. Furthermore, a sequence of strings $s_1, s_2, ..., s_k, ...$ is *monotone increasing* iff for all k, $s_k \prec s_{k+1}$. For example: $1, 00, 01, 000$ is a monotone increasing sequence as $1 \prec 00$, $00 \prec 01$, and $01 \prec 000$.

   (a) (2 points) Prove that every infinite sequence of strings contains a subsequence which is infinite and monotone increasing.

   (b) (2 points) Use this to prove that for every infinite recognizable language L in $\{0,1\}^*$, there is an infinite *decidable* language $L' \subseteq L$. This would, for example, imply that a non-looping Turing machine exists that can accept infinitely many (though not all) instances of the Halting Problem, while rejecting all other strings.
   Hint: The section in Sipser on enumerators (pp.152-153) should be useful here.

3. (3 points) In class you saw a trivial streaming algorithm using $O(n \cdot (\ell + \log n)) \leq O(n \cdot \ell)$ bits of memory for computing the most frequent element (MFE) in an n-element stream with elements from $\{0,1\}^\ell$. But a streaming algorithm only makes one pass over the input. Can you do better with multiple passes? Between passes, it is acceptable to do arbitrary computations on the current memory (but not to read any other input). You may assume for this question that $2^\ell > n^2$.

   (a) (1 point) Prove that with 2^ℓ passes over the data, you can compute the most frequent element using only $O(\ell)$ bits of memory.
   Hint: Have a dedicated pass for each element.

   (b) (1 point) Prove that with p passes over the data (for $p \in \{1, \ldots, 2^\ell\}$), you can compute the most frequent element using only $O(\frac{2^\ell}{p} \cdot \ell)$ bits of memory.
   Hint: Generalize your solution to (a).

   (c) (1 point) Prove that with $2p$ passes over the data (with p in the same range) you can compute the most frequent element using only $O(\frac{n}{p} \cdot \ell)$ bits of memory.