CS 4820, Summer 2010 Out: August 03. **Due: Friday, August 06, 8:30am**

Homework 8

Reading: About Turing machines from lectures, 8.1–4, 8.6–7.

Problem 1. Let us define two languages as follows.

$$L_{ne} = \{\langle M\rangle : \ M \text{ is a Turing machine which accepts at least one string, that is } L(M) \neq \emptyset\},$$
$$L_e = \{\langle M\rangle : \ M \text{ is a Turing machine which does not accept any string, that is } L(M) = \emptyset\}.$$

Note: If you were wondering about the terminology, L_{ne} stands for Language "non-empty", and L_e stands for Language "empty". Note that as mentioned in the lecture, $\langle M\rangle$ stands for a string representation of machine M. Prove that

(a) L_{ne} is undecidable, that is L_e is not recursive. That is, there does not exist any Turing machine N such that N always halts (on any input) and for inputs in L_{ne}, it ends up in the accepting state.

(b) L_{ne} is Turing-recognizable. For this, give a description of a Turing machine N such that $L(N) = L_{ne}$. Note that the machine N is not required to halt on inputs that are not in L_{ne}.

(c) L_e is not recognizable by any Turing machine (not recursively enumerable). That is, there is no Turing machine M that is guaranteed to halt in accepting state when started with any string in L_e.

You might find it helpful to use part (a) and (b) for this part. Note that $L_e \cup L_{ne} \neq \Sigma^*$, since there might be strings that don't describe any Turing machine.

Problem 2. Suppose that someone gives you a black-box algorithm A that takes an undirected graph $G = (V, E)$, and a number k, and behaves as follows:

- If G is not connected, it simply returns "G is not connected".
- If G is connected and has an independent set of size at least k, it returns "yes".
- If G is connected and does not have an independent set of size at least k, it returns "no".

Suppose that the algorithm A runs in time that is polynomial in size of G and the size of k.

Show how, using calls to A, you could then solve the Independent Set Problem in polynomial time: Given an arbitrary graph G, and a number k, does G contain an independent set of size at least k?

Problem 3. Your friends' pre-school-age daughter Madison has recently learned to spell some simple words. To help encourage this, her parents got her a colorful set of refrigerator magnets featuring the letters of the alphabet (some number of copies of the letter 'A', some number of copies of the letter 'B', and so on), and the last time you saw her the two of you spent a while arranging the magnets to spell out words that she knows.

Somehow with you and Madison, things always end up getting more elaborate than originally planned, and soon the two of you were trying to spell out words so as to use up all the magnets in the full set — that is, picking words that she knows how to spell, so that once they were all spelled out, each magnet was participating in the spelling of exactly one of the words. (Multiple copies of words are okay here; so for example, if the set of refrigerator magnets includes two copies of each of C, A, and T, it would be okay to spell out CAT twice.)

This turned out to be pretty difficult, and it was only later that you realized a plausible reason for this. Suppose we consider a general version of the problem of Using Up All the Refrigerator Magnets, where we replace the English alphabet by an arbitrary collection of symbols, and we model Madison's vocabulary as an arbitrary set of strings over this collection of symbols. The goal is the same as in the previous paragraph.

Prove that Using Up All the Refrigerator Magnets is NP-complete.