1. Imagine you are working for a company that has implemented a program. Let’s call that program $P_1$. You are hired as a summer intern to design a new program that is equivalent but runs even faster. Let’s call your new program $P_2$. Your boss says that before you get paid, you must design a program to demonstrate $P_1$ and $P_2$ are equivalent. More formally, you are to design a new program $Q$ which takes two arbitrary programs (i.e., Turing machines) $M_1$ and $M_2$ as input. $Q$ must decide if, for all inputs $x$, the output of $M_1$ on input $x$ equals the output of $M_2$ on input $x$.

Explain why this internship will not be successful!

2. (a) Let $Z^+ = \{0, 1, \ldots \}$. Describe a nondeterministic Turing machine to decide the following language:

$$L_1 = \left\{ x_1 \# x_2 \# \ldots \# x_n : n \in Z^+ \text{ and } \exists \varepsilon_1, \ldots, \varepsilon_n \in \{-1, +1\} \text{ such that } \sum_{i=1}^{n} \varepsilon_i x_i = 0 \right\}.$$  

You may assume that the $x_i$’s are integers.

(b) Describe a nondeterministic Turing machine to recognize the following language:

$$L_2 = \{ \langle M \rangle : M \text{ is a TM and } M \text{ halts on some input} \}.$$  

3. Let $M'$ be a TM that always halts and $L(M') \neq \Sigma^*$. Let

$$L_M' = \{ \langle M \rangle : M \text{ is a TM and } L(M) \not\subseteq L(M') \}.$$  

Show that $L_M'$ is undecidable.

4. Let $A$ and $B$ be two disjoint languages over the alphabet $\Sigma$. Say that language $C$ separates $A$ and $B$ if $A \subseteq C$ and $B \subseteq \overline{C}$. Show that any two disjoint co-recognizable languages are separable by some decidable language. (A language $A$ is said to be co-recognizable if its complement, namely $\overline{A}$, is recognizable.)

5. OPTIONAL BONUS QUESTION: In class, we said that “$A$ is reducible to $B$” (written $A \leq_T B$) if there is a Turing machine that can decide $A$ if it is also given as input a Turing machine that decides $B$. This sort of reduction is called a Turing reduction.

There is a more restrictive type of reduction called a mapping reduction; see Sipser’s Definition 5.20. This type of reduction is written $A \leq_m B$. Roughly speaking, it means that there is a function $f : \Sigma^* \rightarrow \Sigma^*$ that can be computed by a Turing machine such that

$$x \in A \iff f(x) \in B.$$  

It is easy to see that if $A \leq_m B$ (or if $A \leq_m \overline{B}$) then $A \leq_T B$. So Turing reductions are at least as powerful as mapping reductions.

In this question, we will establish that Turing reductions are strictly more powerful. Show that there exist languages $A$ and $B$ such that $A \leq_T B$ but $A \not\leq_m B$ and $A \not\leq_m \overline{B}$. 