Homework 5 – Due Friday, March 8 at 11:59 PM

Reminder Collaboration is permitted, but you must write the solutions *by yourself without assistance*, and be ready to explain them orally to the course staff if asked. You must also identify your collaborators and write "Collaborators: none" if you worked by yourself. Getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden. Collaboration is not allowed on problems marked "INDIVIDUAL."

Problems There are 5 required problems and one bonus problem. Problem 1 will have its own dropbox on Gradescope where you can submit pseudocode and code as plain text files. Problems 2-5 can be submitted as a single PDF as usual.

1. (**Programming TMs**) Construct Turing machines that decide the following languages. That is, the machines should always halt after a finite number of steps on every input, and accept a string w if and only if w is in the given language. Implement your TMs in the following environment: `http://morphett.info/turing/turing.html`. Your solution should contain:

   (i) An implementation-level description of your code.

   (ii) Code that we can copy from your submission and run directly on that website. (Please add comments and make your code as readable as possible.)

   Your TM should enter a state that is clearly marked "accept" or "reject" when it is done. The final tape content does not matter; it does not need to draw an emoji or anything else on its tape.

   (a) $L_1 = \{w \in \{0,1\}^* \mid$ Every even position of w is a $0\}$.

   Hint: You could solve this problem with a DFA. How would you simulate that DFA using a TM?

   (b) $L_2 = \{0^n 1^m \mid m, n \geq 0$ and m is a multiple of $n\}$.

2. (**Recognizability vs. Decidability**) Recall the high-level description of a TM recognizer for the language $\{\langle p \rangle \mid p$ is a k-variate integer polynomial and there exists $x_1, \ldots, x_k$ such that $p(x_1, \ldots, x_k) = 0\}$ that we described in class.

   Input : Encoding of k-variate polynomial p
   1. For every possible setting of $x_1, \ldots, x_k$ to integer values:
   2. Evaluate $p(x_1, \ldots, x_k)$. If it equals 0, *accept*.

   Explain in a few sentences what is **wrong** about the following attempt to construct a TM decider for the same language:

Input : Encoding of k-variate polynomial p

1. For every possible setting of $x_1, \ldots, x_k$ to integer values:
2. Evaluate $p(x_1, \ldots, x_k)$.
3. If any evaluation equals 0, *accept*. Otherwise, *reject*.

3. (**Closure properties**)

   (a) Let Σ be a finite alphabet. For a string $w = w_1 \ldots w_n$, where each $w_i \in \Sigma$, define $\text{Rep}(w) = w_1 w_1 w_2 w_2 \ldots w_n w_n$. That is, $\text{Rep}(w)$ is the string obtained by repeating each character in w one additional time. Given a language $L \subseteq \Sigma^*$, define the language $\text{Rep}(L) = \{\text{Rep}(w) \mid w \in L\}$.

   Suppose L is decided by a Turing machine M. Using M as a subroutine, give an implementation-level description of a Turing machine N that decides $\text{Rep}(L)$. Briefly explain in English what N does and why it works. It may be useful to use a multi-tape TM to solve this problem.

   (b) Are the decidable languages closed under the Rep operation? Are the Turing-recognizable languages closed under the Rep operation? Explain your answers.

4. (**Insert-only TM**) An *insert-only* Turing machine (ITM) is the same as a basic (deterministic) one-tape Turing machine, but instead of writing a new symbol to the current cell under the tape head, it inserts a new cell with that symbol to the immediate left of the current cell. It can also move the tape head while leaving the tape content unchanged.

   Here are some examples of how an ITM could work.

   - If an ITM is in configuration $867p309$ and its transition function specifies $\delta(p, 3) = (q, 5, \texttt{R})$, then the next configuration will be $86753q09$. (The ITM reads symbol 3, inserts symbol 5, then moves to the right.)
   - If an ITM is in configuration $867r309$ and its transition function specifies $\delta(r, 3) = (s, 5, \texttt{L})$, then the next configuration will be $867s5309$. (The ITM reads symbol 3, inserts symbol 5, then moves to the left.)
   - If an ITM is in configuration $867s5309$ and its transition function specifies $\delta(s, 5) = (t, *, \texttt{L})$, then its next configuration will be $86t75309$. Here, "writing" the $*$ symbol indicates that the TM should move without modifying the tape.

   (a) Modify Definition 3.3 in Sipser to give the formal definition of a insert-only TM. Most of it should stay the same, but pay special attention to the syntax of the transition function (item 4), which should handle the special $*$ symbol that is not part of the tape alphabet.

   (b) Show that insert-only TMs are *no more* powerful than basic TMs. That is, use an implementation-level simulation argument to show that every language recognizable by an insert-only TM is also Turing-recognizable.

   (c) Show that insert-only TMs are *at least* as powerful as basic TMs. That is, use an implementation-level simulation argument to show that every Turing-recognizable language is also recognizable by an insert-only TM.

   Hint: You may want to enlarge the tape alphabet of your ITM to include another special symbol that signals it to ignore some of the tape content.

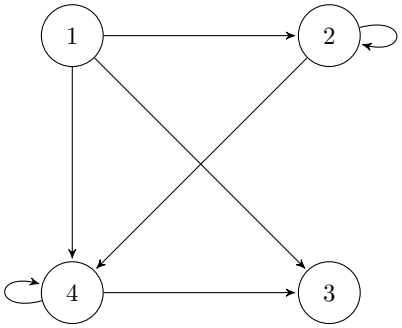
Parts (a) and (b) together show that insert-only TMs are equivalent to basic TMs: They exactly recognize the class of Turing-recognizable languages.

5. (**Nondeterministic TMs**)

   (a) To present a (directed) graph G as input to a Turing machine, one needs to provide a string *encoding* it, denoted by $\langle G \rangle$. (You can think of $\langle G \rangle$ as the result of applying a "toString" method for the object G.) One convenient such encoding is the flattened adjacency matrix representation: A graph G on vertices $1, 2, \ldots, n$ can be encoded as $\langle G \rangle = \#w_1\#w_2\#\ldots\#w_n\#$ where each $w_i \in \{0,1\}^n$ is a string such that $w_{i,j} = 1$ if there is an edge from i to j in G, and $w_{i,j} = 0$ if there is not an edge.

      i. Draw the directed graph on 3 vertices that is encoded by #100#110#001#.

      ii. What is the encoding $\langle H \rangle$ of the following graph H?



   (b) A (directed) *triangle* in a directed graph consists of three vertices i, j, k, such that there is an edge from i to j, an edge from j to k, and an edge from k to i. Thus, a digraph G contains a triangle if and only if there exist i, j, k such that $w_{i,j} = w_{j,k} = w_{k,i} = 1$ in its encoding.

   Give an implementation-level description of a **nondeterministic** (multi-tape) TM deciding the language $L = \{\langle G \rangle \mid \text{there exists a triangle in directed graph } G\}$. Briefly explain why your construction works. It is, of course, possible to solve this problem with a deterministic TM, but we'd like you to practice using nondeterminism, so your solution should make use of the ability to "nondeterministically guess" in a meaningful way.

   Hint: Recall that a nondeterministic TM is a decider if it halts on every input, on every computation branch.

6. (**Bonus problem**) Let A be a Turing-recognizable language which is not decidable. (We will prove later in the course that such languages exist.) Consider a TM M that recognizes A. Prove that there are infinitely many input strings on which M loops forever. If you need to construct a TM to solve this problem, you can give a high-level description.