University of Virginia - cs3102: Theory of Computation	Spring 2010
PS4	**Due: 23 March 2010 (2:00pm)**

This problem set covers material through the end of Chapter 3. Answer all five questions. For full credit, answers must be concise, clear, and convincing, not just correct. Note that you only have *one week* for this problem set, please get started on it early.

You are strongly encouraged to use LaTeX to produce your submission, and we have provided a LaTeX template to assist with this. You may, however, handwrite your answers *so long as your writing is legible and easily interpreted.* For full credit, answers must be concise, clear, and convincing, not just correct.

Collaboration Policy. For this assignment, we will follow the same collaboration and resource policy as on Problem Sets 1-3. See the Problem Set 1 handout for a full description of the policy. Use collaboration to help you learn, but not in ways that prevent you from learning to solve problems on your own.

Problem 1: Designing Turing Machines. Design a deterministic Turing Machine that decides the language $\{0^n1^n2^n \mid n \geq 0\}$. You should provide descriptions of your Turing Machine at the three levels of detail described in Section 3.3:

a. Provide a *high-level description* of your machine.

b. Provide an *implementation description* of your machine.

c. Provide a full *formal description* of your machine (this is tedious, but everyone should do it once!). You may use any TM simulator you want to develop and test your solutiemon, but it is not necessary to do so. One good TM simulator is JFLAP: http://www.jflap.org/jflaptmp/. But be careful, it simulates a doubly-infinite tape TM (see Problem 4), not the standard TM we defined.

Problem 2: Deciding *SQUAREFREE.* Provide an *implementation description* of a Turing Machine that decides the language of squarefree sequences in $\{a, b, c\}^*$ (from Exam 1). A *squarefree* sequence is a sequence that contains no adjacent repeating subsequences of any non-zero length. For example, ab and $abcba$ are squarefree, but aa is not since it contains a^2 and $abcbacbab$ is not since it contains $(cba)^2$. Your description can be at a high-level, but should be detailed enough to convince a skeptical reader that there is a TM that decides *SQUAREFREE.* (For clarity of presentation, you may want to start by providing a high-level description.)

Problem 3: Equivalence of 2-DPDA+ϵ and Turing Machine. In Class 14, we informally argued that a 2-stack deterministic pushdown automaton (with forced ϵ-transitions) is equivalent to a Turing Machine. For this problem, prove one direction of the equivalence: that a 2-DPDA+ϵ can simulate any Turing Machine. (Hint: your proof should show how to construct a 2-DPDA+ϵ from a given Turing Machine. For a clear proof, you will need to define a formal notation for describing a 2-DPDA+ϵ.)

Problem 4: Robustness of Turing's Model. (Sipser problem 3.11) A *Turing machine with doubly infinite tape* is similar to an ordinary Turing machine, but its tape is infinite to the left as well as to the right. The tape is initially filled with blanks except for the portion that contains the input. Computation is defined as usual except that the head never encounters an end to the tape as it moves leftward. Show that the class of languages recognized by a Turing machine with doubly infinite tape is equivalent to the class of languages recognized by a regular Turing machine.

Problem 5: Deciders and Recognizers. In Class 15, we argue that DFAs, DPDAs, and NFAs always terminate, hence the sets of languages that can be recognized by these machines are equivalent to the sets of languages that can be decided by them. By contrast, a Turing Machine may run forever on some inputs, so there may be languages which can be recognized but not decided by a Turing Machine (we will see examples of such languages next week). What about nondeterministic pushdown automata?

a. Draw a Venn diagram including these sets:

   - *DPDA-Decidable* (DPDA-D): languages that can be *decided* by a deterministic PDA.
   - *NPDA-Decidable* (NPDA-D): languages that can be *decided* by a nondeterministic PDA.
   - *TM-Decidable* (TM-D): languages that can be *decided* by a Turing Machine.
   - *NPDA-Recognizable* (NPDA-R): languages that can be *recognized* by a nondeterministic PDA.
   - *TM-Recognizable* (TM-R): languages that can be *recognized* by a Turing Machine.

b. Prove that your diagram correctly depicts the relationship between *NDPDA-Recognizable* and *TM-Decidable*.

c. Prove that your diagram correctly depicts the relationship between *NDPDA-Recognizable* and *NDPDA-Decidable*. (For example, if your diagram showed them as the same circle, prove that the sets are equivalent. If your diagram shows one set inside the other, prove that all languages in one set are in the other set but that there is some language in the outer set that is outside the inner set.)