# CS310 2021 (Automata Theory)
# Exercise problem set 7: Turing machines

1. Give a high-level description of a Turing machine that recognizes the following language. (You may use the TM defined for one subpart as a subroutine for another subpart of the question, see Section 8.3.3 of Hopcroft Ullman for an example of this). Let the input alphabet be $\Sigma = \{0, 1, \#\}$ and let $w \in \{0,1\}^*$ be a given word.
   (a) Given a word $w\#$ on the tape, generate $w\#0^{|w|}$ on the tape.
   (b) Given $w\#$ on the tape, generate $w\#w$ on the tape.
   (c) Given $w\#$ on the tape, generate $w\#w^R$ on the tape, where $w^R$ is the reverse of the word $w$.
   (d) Given $w\#$ on the tape, generate $w^R\#$ on the tape.

2. Give the full formal description (using the state diagram or by describing the transition function $\delta$ ) of a deterministic TM that *decides* the language $L = \{0^n1^n2^n \in \{0,1,2\}^* \mid n \geq 1\}$.

3. While simulating multi-tape Turing machines with a single-tape Turing machine as done in class, we used an operation which allowed us to move the entire tape content from a point, to one cell right, with a new blank at this point. Give a formal description how the Turing machine would perform this operation. Note that after shifting right, the head must return to the same point.

4. Design a deterministic Turing machine for the language $\{ww \mid w \in \{0,1\}^*\}$ with (a) multiple tapes and (b) single tape. A high-level description of the TM will suffice.

5. Prove that $L$ is decidable iff there exists a non-deterministic Turing machine that decides it. Write the proof in full detail.

6. Prove or disprove whether Turing-recognizable languages are closed under the following operations:
   (a) Union
   (b) Intersection
   (c) Complementation

(d) Star

(e) Concatenation

If your answer is that they are closed for a particular operation, you must prove it formally. For instance, for Union, you need to take two Turing recognizable languages, i.e., take two Turing machines recognizing each language, and show that you can design a Turing machine that accepts their union. If your answer is that they are not closed for a particular operation, you must give a counter-example.

7. Let $k$-PDA be a pushdown automaton that has $k$ stacks. Thus a 0-PDA is an NFA and a 1-PDA is the usual PDA. Recall that we say two models are equivalent if they accept the same set of languages. Prove or disprove the following. For either you must give explicit proofs/counterexamples.

   (a) 0-PDA and 1-PDA are not equivalent.

   (b) 1-PDA and 2-PDA are equivalent.

   (c) 2-PDA and 3-PDA are equivalent.

   Can you compute the minimum $k$ such that $k$-PDA are equivalent to Turing machines?

8. A *queue* automata is like PDA but it has a stack replaced by a queue.We define acceptance by final state, i.e., when it reaches a special accept state it accepts.

   (a) Formally define the model of queue automata (i.e., as a n-tuple etc).

   (b) Prove or disprove: $L$ can be accepted by a queue automaton iff $L$ is Turing recognizable.

9. An *enumerator* $E$ is like a Turing machine but with a printer, which is used to print output strings. It has a finite control, an infinite work-tape and the printer. It starts with a blank input on its work tape. The language *enumerated* by this machine is the set of all strings that it eventually prints out. Note that if it doesnt halt, it can print an infinite set of strings so its language can be infinite. Moroever it can generate strings in any order, possibly with repetitions.

   (a) Formally define the model of enumerator. You can think of it as a two-tape machine which uses its second tape as the printer.

   (b) Prove or disprove: $L$ is enumerated by some enumerator $E$ iff $L$ is Turing recognizable.