Problem 1

Consider the following Turing machine variant: a two-layered Turing machine is a Turing machine whose input tape has two symbols in each cell, a top symbol and a bottom symbol. A two-layered Turing machine transitions like a regular Turing machine, except that at each step, it simultaneously reads both the top and bottom symbols in the current cell, and likewise simultaneously updates both the top and bottom symbols in the current cell. (Different values can be written to the top and bottom symbols.) The input to a two-layered Turing machine appears on the bottom of cells on the tape starting from the left; all the top symbols (as well as the bottom symbols beyond the input cells) are initially empty.

a. Show that any language recognized by a two-layered Turing machine can also be recognized by a regular Turing machine.

b. How could you generalize part (a) for a Turing machine variant with an $n$-layered input tape (which has $n$ symbols in each cell)?

The following questions are lab problems.

Lab Problem 1

A palindrome is a string $w$ over the alphabet $\Sigma = \{0, 1\}$ with the property that $w = w^R$, where $w^R$ is the reverse of $w$. For example, the strings $\varepsilon$ (the
empty string), 010, and 0110 are palindromes.

a. Give a high level description of a Turing machine that recognizes the language of palindromes. Some questions that may be helpful to consider: how will the Turing machine remember the work that has been done so far? What tape alphabet will it use?

b. Draw the state diagram for your Turing machine.

c. Is the language of palindromes regular? Prove your answer.

Lab Problem 2

John Cena wants to design a new model of computation called Windows Vista. A Windows Vista program is a finite sequence of lines, each containing exactly one command. When running a Windows Vista program, Windows Vista reads the lines one at a time and keeps track of two stacks $A$ and $B$, as well as a symbol $x$. As in the case of Turing machines, the input to a Windows Vista program is a string over some alphabet $\Sigma$ that does not contain the special symbol $\omega$. The symbol $x$, as well as the symbols in the stacks $A$ and $B$, are members of a larger alphabet $\Gamma$, where $\Sigma \subseteq \Gamma$ and $\omega \in \Gamma$. The input to the Windows Vista program is initially stored in stack $B$ with the first character on top, while stack $A$ is initially empty, and $x$ is initially equal to $\omega$.

John permits the following commands in a Windows Vista program:

- **push $A$:** Push $x$ onto stack $A$.
- **push $B$:** Push $x$ onto stack $B$.
- **pop $A$:** Replace $x$ with the top symbol of $A$, and delete the top of $A$.
- **pop $B$:** Replace $x$ with the top symbol of $B$, and delete the top of $B$.
- **set $<\text{symbol}>$:** Set $x = <\text{symbol}>$ (where $<\text{symbol}> \in \Gamma$).
- **if $<\text{symbol}>$:** Unless $x = <\text{symbol}>$, skip the following line.
- **goto $<\text{string}>$:** If there is a label $<\text{string}>$ line in the program, go there. If there is none or there is more than one, ignore this command.
- **label $<\text{string}>$:** Ignore and move to the next line.
Note that:

- If `pop` is called on an empty stack, then \( x \) is replaced with the special symbol \( \sqcup \).
- If Windows Vista were to read past the last line of the program, it halts.

A Windows Vista program accepts its input string if the program halts with \( x \neq \sqcup \). The language of a Windows Vista program \( Y \) is the set of strings \( s \in \Sigma^* \) such that \( Y \) accepts \( s \).

Prove that Windows Vista is equivalent to a Turing machine in the following sense:

a. Every Turing machine can be converted to an equivalent Windows Vista program (i.e., a Windows Vista program that accepts exactly the same strings that the Turing machine does).

b. Every Windows Vista program can be converted to an equivalent Turing machine.

**Hint:** For either part, if it is more convenient, you may choose to use a non-standard Turing machine (e.g., multi-tape, doubly-infinite tape, or some other equivalent kind of Turing machine). You don’t have to prove that the kind of Turing machine you use is equivalent to a standard Turing machine, as long as equivalence has been shown in class or in the book.

**Lab Problem 3**

Turing machine! Y u no move left? In this problem, you will examine two Turing machine variants. There is no need to provide a formal (low-level) description of any Turing machines you construct; high-level descriptions will suffice.

a. A Turing machine with `left-reset` instead of `left` is defined as a Turing machine that, rather than moving either left or right at each step, can either move its head one space to the right or jump all the way to the start of the tape (on the left). Show that such a Turing machine can simulate a regular Turing machine and show the other direction as well.
b. A Turing machine with *stay-put* instead of *left* is defined as a Turing machine that, rather than moving either left or right at each step, can either move its head one space to the right or not move at all. Prove that these machines can only recognize regular languages.