Problem 1

Walter Chestnutt is suspicious that rival farmers have been stealing his prize-winning turmeric plants. The midterm elections finished last week, but the votes were so close that the political bosses of Yammany Hall are demanding a recount. Now that he’s running for public office, it’s especially important to show that he’s contributing his share of root vegetables to society and the market as a whole.

Every year, he obtains two seeds from each of his turmeric plants and plants those for the next year, thus doubling his number of plants. Because he is an extremely talented farmer, every seed he plants yields delicious crops. To check whether his turmeric is being stolen, Walter wants to determine whether the number of his plants is a power of two. If it is not, then he knows that some of his plants have been stolen.

To do so, he wants to design a Turing machine that will decide $L$, where $L = \{1^{2^n} \mid n \geq 0\}$ over the alphabet $\Sigma = \{1\}$.

a. Help Walter give a formal diagram of a Turing machine that recognizes $L$, with all of its states and transitions. Include an informal description of what your Turing machine does.

b. Analyze the runtime of your TM, in big theta ($\Theta$) notation.
The following questions are lab problems.

Lab Problem 1

Ginger’s friend Ginny Tseng convinces her to try online dating—after all, it’s how Ginny met Cole Robby. Ginger decides to look for a date on JicaMate. She doesn’t want to let her family know anything about her search, though, as they think she should be focusing on her farmwork. Luckily, they’re bored to tears when they see text that’s just 0, 1, and \_. It doesn’t help that many of the O’Nions have chronically watery eyes and generally poor eyesight.

a. Ginger has come up with a clever plan to protect her messages: instead of using the English alphabet, she will use the alphabet \{0, 1\}. To put her plan into action, she needs to convince herself that she can encode any message into an encrypted form. She knows that two Turing machines \(T_1, T_2\) with different input alphabets \(\Sigma_1, \Sigma_2\) are equivalent if and only if there is some function \(f : \Sigma_1^a \rightarrow \Sigma_2^b\) such that \(f\) is invertible (essentially, \(f\) is a lookup table where each input has a unique output) and for every string \(x\) that is accepted, rejected, or looped forever on by \(T_1, T_2\) similarly accepts, rejects, or loops forever on input \(f(x)\). Help Ginger by showing how any Turing machine (with original input alphabet \(\Sigma\)) can be converted into an equivalent Turing machine with input alphabet \{0, 1\}.

**Hint:** the tape alphabet does not need to be \{0, 1, \_\}.

b. Ginger has found a promising match in a guy named Rudy, but she needs to keep her thoughts on him secret from her family. Describe how any Turing machine (with original tape alphabet \(\Gamma\)) can be converted to a Turing machine with tape alphabet \{0, 1, \_\} such that the languages of both are the same. You may assume that a state transition can move the head to left or right (as in a standard TM) and can also choose to keep the head stationary on the tape.

c. All this planning has gotten Ginger interested in what else can be done with only a binary choice. She’s especially intrigued by JicaMate’s algorithms, which somehow condense the problem of finding a soulmate to answering yes/no questions. Describe how any nondeterministic Turing machine (such as finding a soulmate) can be converted into a nondeterministic Turing machine where any \((state, symbol)\) tuple has at most two transitions. You may assume that a state transition can choose to keep the head stationary on the tape.
d. Justify why the choices made by a nondeterministic TM can be encoded as a binary string.

Lab Problem 2

In this problem, you will give a high level description of a Turing machine that recognizes the following language. Let $\Sigma = \{0, 1, \times, =\}$. Then:

$$L = \{ w \mid w \text{ is of the form } "x \times y = z" \text{ with } x, y, z \in (0 \cup 1)^* \text{ such that the product of } x \text{ and } y \text{ is } z \text{ when interpreted as binary integers} \}$$

a. First, explain the operation of incrementing and decrementing. That is, if there is a string $x \in (0 \cup 1)^*$ on the tape of your TM, give a high level description of how you would increment and decrement $x$ by one (for decrementing, ignore the case where $x = 0$).

b. Now give a high level description of the TM that recognizes $L$.

Note: There are multiple ways of doing binary multiplication, some of which have better runtimes than others. You will not be graded on the runtime of your TM—only worry about correctness. You can just say ‘increment’ and ‘decrement’ without re-explaining the process. It may also be useful to consider a multi-tape Turing machine.

Lab Problem 3

Soon after joining JicaMate, Ginger gets a response from Rudy. Ginny Tseng advises her to be honest, so Ginger admits to Rudy that she needs to keep their communications secret. He’s impressed by her knowledge of Turing machines, and they immediately get along. They decide to work together on a project as a bonding experience before they meet up.

They want to design a new kind of Turing machine, called the Yamputer, in which a program is a finite sequence of lines, each containing exactly one command. The interpreter reads the lines one at a time, and keeps track of two stacks, $A$ and $B$, as well as the current symbol $x$. $A$ and $B$ are strings over the tape alphabet $\Gamma$, and $x \in \Gamma$. As in the case of Turing machines, the input string is over a smaller alphabet $\Sigma$. $\Gamma$ contains a special symbol $\omega$, and $\Sigma$ does not. The input is initially stored in stack $B$ with the first character on top, while stack $A$ is initially empty, and $x$ is initially equal to $\omega$. 
Ginger and Rudy permit the following commands in the 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 <symbol>**: Set $x = <symbol>$.
- **if <symbol>**: Unless $x = <symbol>$, skip the following line.
- **goto <string>**: If there is a label <string> line in the program, go there. If there is none or there is more than one, ignore this command.
- **label <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 $\underline{\_}$.
- If the Yamputer were to read past the last line of the program, it halts.

When (if) the program halts, if $x = \underline{\_}$ then the original string is considered rejected; otherwise it is considered accepted. The language of a Yamputer program $Y$ is the set of strings $s \in \Sigma$ such that $Y$ accepts $s$.

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

a. Every Turing machine can be converted to an equivalent Yamputer program.

b. Every Yamputer 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.