Problem 1

Provided below is a Turing machine for the language $\{a^n b^n \mid n \in \mathbb{N}_0\}$. Note that $B$ is being used for a blank, and the input alphabet is $\{a, b\}$.

Modify the above to create a TM for the language $\{a^n b^{2n} \mid n \in \mathbb{N}_0\}$, and draw a diagram for the resulting TM.

Add a new state, $q_5$, to handle erasing an extra $b$ for each $a$. The transition from $q_2$ to $q_3$ is deleted, and two new transitions are added: $\delta(q_2, b) = (q_5, \square, L)$ and $\delta(q_5, b) = (q_3, \square, L)$.
Problem 2

Each of the “proofs” below claims to prove a conjecture. The conjectures might be true or false, but both of the proofs are bad. For each proof, explain clearly why the provided proof does not satisfactorily prove the given conjecture.

a. **Conjecture**: The “BusyBee” language, defined below, is undecidable:

\[ L_{\text{BusyBee}} = \{ \langle M, w, k \rangle \} \]

where \( M \) describes a Turing machine and \( k \) is the number of different states \( M \) enters before halting on input \( w \). (Note that \( q_{\text{accept}} \) and \( q_{\text{reject}} \) are counted toward this total.)

**Proof by reduction**: We prove that \( L_{\text{BusyBee}} \) is undecidable by reducing \( \text{HALT}_{\text{TM}} \) to it. Construct \( M_{\text{BusyBee}} \) given a machine \( M_{\text{HALT}} \) that decides \( \text{HALT}_{\text{TM}} \).

\[ M_{\text{BusyBee}} = \]

Simulate \( M_{\text{HALT}} \) on input \( \langle M, w \rangle \). If \( M_{\text{HALT}} \) accepts, simulate \( M \) on \( w \). On a second tape, list the states of \( M \) and add a mark on each state that is entered. Count the number of marked states on the second tape. If the number matches \( k \), accept. Otherwise, reject (without simulating \( M \) on \( w \) since it does not halt). Since building \( M_{\text{BusyBee}} \) requires \( M_{\text{HALT}} \), which we know does not exist, this proves that \( L_{\text{BusyBee}} \) is undecidable.

The proof does the reduction in the wrong direction. It shows that you can decide \( L_{\text{BusyBee}} \) if you can decide \( \text{HALT}_{\text{TM}} \) (that is, that \( L_{\text{BusyBee}} \) is not harder than \( \text{HALT}_{\text{TM}} \)), but it does not show the conjecture, which is that \( L_{\text{BusyBee}} \) is undecidable. To show that, we would need a reduction that shows how to solve \( \text{HALT}_{\text{TM}} \) if you were given a machine that can solve \( L_{\text{BusyBee}} \).
b. **Conjecture**: The set of languages that can be recognized by a deterministic pushdown automaton is equivalent to the set of languages that can be recognized by a Turing machine.

**Proof by simulation**: We prove the conjecture by showing how to simulate any DPDA with a TM... (*unnecessary details elided*)... Thus, we can simulate a DPDA with a TM. This proves a DPDA is equivalent to a TM.

The proof only covers one direction: It shows a DPDA can be simulated by a TM. To show the sets of languages they recognize are equivalent, we need to show both directions. (In this case, this would be impossible, since it’s not possible for a DPDA to simulate a TM.) Hence, what is actually proven here is that a DPDA is no more powerful than a TM, which is not surprising since the set of languages that can be recognized by a DPDA is a subset of the Turing-recognizable languages.
Problem 3

Consider the Turing machine $M = (\{q_0, q_1, q_f\}, \{0, 1\}, \{\theta, 1, \square\}, \delta, q_0, q_{accept}, q_{reject})$, where $\delta$ consists of the following rules:

\[
\begin{align*}
\delta(q_0, \square) &= (q_{reject}, \square, R) \\
\delta(q_0, \theta) &= (q_0, \square, R) \\
\delta(q_0, 1) &= (q_1, \square, R) \\
\delta(q_1, \square) &= (q_{accept}, \square, R) \\
\delta(q_1, \theta) &= (q_{reject}, \square, R) \\
\delta(q_1, 1) &= (q_1, \square, R)
\end{align*}
\]

a. Informally but clearly describe the language $L(M)$.

$L(M) = \{\theta^*1^+\}$, that is, the language consisting of zero or more $\theta$s followed by at least one $1$s.

b. Give a string in this language that is of length at least 4, and show the sequence of moves of this TM when this string is the input to $M$.

Consider the string $s = \theta 111$. The following sequence of moves recognizes $s$:

$q_0 \theta 111$
$\square q_0 111$
$\square \square q_1 1$
$\square \square \square q_1 \square$
$\square \square \square \square q_{accept} \square$
Problem 4

Answer each of the following as indicated.

a. Indicate whether the following is true or false and fully justify your answer: If language $A$ is a subset of language $B$, and $A$ is Turing-recognizable, then $B$ is Turing-recognizable.

*False.* Let $A = \emptyset$, the empty set. Then $A$ is Turing-recognizable – it’s accepted by a TM that rejects everything – but $A \subseteq B$ for any language $B$. But we know that there are languages that are not Turing-recognizable.

b. Indicate whether the following is true or false and fully justify your answer: A nondeterministic TM accepts a string $w$ if and only if every possible computation on the input leads to an accepting state.

*False.* A non-deterministic TM accepts $w$ if at least one computation on $w$ leads to an accepting state.

c. Indicate whether the following is true or false and fully justify your answer: The computational path of a TM on an input $w$ always either halts accepting, halts rejecting, or enters a never-halting loop.

*True.* The only two ways for a TM to halt are by accepting and rejecting. If the TM does not reject or accept, it never halts, i.e., enters a never-halting loop.

d. Circle the number to the left of each of the languages below that is Turing-decidable but not context-free:

1. $\{ww^R \mid w \in \{a, b\}^+\}$
2. $\{wcw \mid w \in \{a, b\}^+\}$
3. $\{a^{2n}b^n c^{2n} \mid n \in \mathbb{N}_0\}$

e. Circle the number to the left of each of the languages below that is Turing-recognizable but not Turing-decidable:

1. $\{\langle M, w \rangle \mid \text{TM } M \text{ accepts input } w\}$
2. $\{\langle M \rangle \mid L(M) \neq \emptyset\}$
3. $\{\langle M_1, M_2 \rangle \mid L(M_1) = L(M_2)\}$

f. Circle the number to the left of each of the languages below that is not Turing-recognizable:

1. $\{\langle M, w \rangle \mid \text{TM } M \text{ halts on input } w\}$
2. $\{\langle M \rangle \mid L(M) \neq \emptyset\}$
3. $\{\langle M_1, M_2 \rangle \mid L(M_1) \cap L(M_2) = \emptyset\}$
Problem 5

Determine whether the following problems are decidable or not:

a. Is a number $m$ prime?

This problem is decidable because there is an algorithm which, after a finite number of steps, will halt and give us a definitive answer to the question. Here is the algorithm:

On input $m$:

1. Divide the number $m$ by all the numbers between 2 and $\sqrt{m}$, starting from 2.
2. If any of these numbers produces a remainder zero, reject.
3. Otherwise, accept.

The algorithm is guaranteed to halt with a definitive “yes” or “no” answer because the set of numbers between 2 and $\sqrt{m}$ is finite and therefore bounded.

b. Given a regular language $L$ and a string $w$, is $w \in L$?

Construct a DFA that accepts $L$ and check if $w$ is accepted. The algorithm is guaranteed to terminate in a finite number of steps – namely, $|w|$ since it must read an input character each time – and give a “yes” or “no” answer depending on whether it is in a final state at the end of the computation. So this problem is decidable.
Problem 6

Prove that the language \( L = \{\langle M \rangle \mid M \text{ is a TM and } M \text{ accepts string } w \text{ whenever it accepts } w^R \} \) is undecidable.

As usual, we assume that \( L \) is decidable, and we let \( M_L \) be the Turing machine that decides \( L \). We're going to construct a TM that uses \( M_L \) to decide \( A_{TM} \). Since \( A_{TM} \) is known to be undecidable, this leads to a contradiction, proving that \( L \) is undecidable.

The decider for \( A_{TM} \) can be built as follows:

\( M_{ATM} \) = “on input \( \langle M, w \rangle \):

1. Construct the code for the following Turing machine \( M_1 \):
   \( M_1 = \) ‘on input \( x \):
   a. Simulate \( M \) on \( w \).
   b. If \( M \) rejects \( w \), reject.
   c. If \( M \) accepts \( w \), accept if \( x = 01 \); otherwise, reject.’

2. Simulate \( M_L \) on input \( \langle M_1 \rangle \).

3. If \( M_L \) rejects, accept; otherwise, reject.”

The key point is this: What is the language of \( M_1 \)? When \( M \) accepts \( w \), \( L(M_1) = \{01\} \), which is not in \( L \), since it doesn't include 10. When \( M \) rejects \( w \), \( L(M_1) = \emptyset \), which is in \( L \).