(2 pts) 1. Give a simple verbal description of the function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ computed by the Turing machine below.

```
q0 \rightarrow 1 \rightarrow 0, R
\square \rightarrow 1, R
\rightarrow 0, R
\rightarrow q_{HALT}
```
2 Fermat’s conjecture, until recently one of the most famous unproven statements in mathematics, asserts that no positive integers $a, b, c$ can satisfy the equation

$$a^n + b^n = c^n$$

for any integer $n > 2$. Ignoring the fact that the conjecture has now been proved, explain how a solution to the halting problem would allow you to determine the truth or falsity of Fermat’s conjecture.

3 Prove that Turing-recognizable languages are closed under the prefix operation.
Language $L$ is said to separate languages $L', L''$ if $L' \subseteq L$ and $L'' \subseteq \overline{L}$, as illustrated in the figure below. Prove that any two disjoint languages $L', L''$ whose complements are Turing-recognizable are separated by some decidable language.
For each of the languages below, determine whether it is decidable. Prove your answers.

(3 pts)  a. \( \{D_1, D_2\} : \) the DFAs \( D_1 \) and \( D_2 \) accept a string in common

(3 pts)  b. \( \{G, w\} : \) the CFG \( G \) generates a string that starts with \( w \)

(3 pts)  c. \( \{n\} : \) the decimal expansion of \( \pi \) contains \( n \) consecutive 0s

(3 pts)  d. \( \{M, q\} : \) the Turing machine \( M \) enters the state \( q \) on some input
SOLUTIONS
(2 pts) 1. Give a simple verbal description of the function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ computed by the Turing machine below.

Solution. The Turing machine interprets its input as an integer $n$ in binary, written left to right starting with the least significant bit, and computes the function $n \mapsto n + 1$. 

Fermat’s conjecture, until recently one of the most famous unproven statements in mathematics, asserts that no positive integers $a, b, c$ can satisfy the equation

$$a^n + b^n = c^n$$

for any integer $n > 2$. Ignoring the fact that the conjecture has now been proved, explain how a solution to the halting problem would allow you to determine the truth or falsity of Fermat’s conjecture.

**Solution.** Consider a Turing machine $M$ that ignores its input and proceeds to enumerate all tuples $(a, b, c, n)$ of positive integers in increasing order by $a + b + c + n$, until it finds a tuple with $n > 2$ and $a^n + b^n = c^n$ at which point it enters the accept state. Clearly $(M, \varepsilon) \notin \text{HALT}$ if and only if Fermat’s conjecture is true. Thus, a solution to the halting program would allow us to determine the status of Fermat’s conjecture.

---

Prove that Turing-recognizable languages are closed under the prefix operation.

**Solution.** Let $L$ be a Turing-recognizable language, and let $M$ be a Turing machine that recognizes $L$. Consider the nondeterministic Turing machine $M'$ which on input $w$, nondeterministically guesses a string $u \in \Sigma^*$, runs $M$ on input $wu$, and outputs the result (if $M$ halts on $wu$).

- If $w \in \text{prefix}(L)$, there exists $u \in \Sigma^*$ such that $M$ accepts $wu$. Therefore, $M'$ accepts $w$ on at least one computational path.
- If $w \notin \text{prefix}(L)$, then $M$ rejects all strings of the form $wu$, either explicitly or by failing to halt. Therefore, $M'$ does not accept such $w$ on any computational path.

Thus, $M'$ recognizes $\text{prefix}(L)$. This completes the proof because nondeterministic Turing machines recognize only Turing-recognizable languages.
Language $L$ is said to separate languages $L', L''$ if $L' \subseteq L$ and $L'' \subseteq \overline{L}$, as illustrated in the figure below. Prove that any two disjoint languages $L', L''$ whose complements are Turing-recognizable are separated by some decidable language.

![Diagram](image)

**Solution.** Let $M'$ and $M''$ be Turing machines that recognize $\overline{L'}$ and $\overline{L''}$, respectively. On input $w$, we will run $M'$ and $M''$ in parallel until one of them accepts $w$. This will happen eventually because $L(M') \cup L(M'') = \overline{L'} \cup \overline{L''} = \overline{L'} \cap \overline{L''} = \emptyset = \Sigma^*$. If $M''$ is the first to accept, then $w \notin L''$ and we will accept $w$; if $M'$ is the first to accept, then $w \notin L'$ and we will reject $w$. Either way, the described Turing machine always halts, accepts all of $L'$, and rejects all of $L''$, which means that its language is decidable and separates $L'$ and $L''$. 
For each of the languages below, determine whether it is decidable. Prove your answers.

(3 pts) a. \( \{ D_1, D_2 \} : \) the DFAs \( D_1 \) and \( D_2 \) accept a string in common

(3 pts) b. \( \{ G, w \} : \) the CFG \( G \) generates a string that starts with \( w \)

(3 pts) c. \( \{ n \} : \) the decimal expansion of \( \pi \) contains \( n \) consecutive 0s

(3 pts) d. \( \{ M, q \} : \) the Turing machine \( M \) enters the state \( q \) on some input

---

**a.** Decidable: create a DFA for \( L(D_1) \cap L(D_2) \) using the Cartesian-product construction from class and check whether an accept state is reachable from the start state.

**b.** Decidable. Construct a PDA for \( L(G) \) and a DFA for the regular language \( w \Sigma^* \); then combine them into a PDA for \( L(G) \cap w \Sigma^* \) using the Cartesian-product construction from class; finally, convert the resulting PDA into a grammar and test whether the grammar generates any strings (using the algorithm from class).

**c.** Decidable. Let \( L \) denote the language in question. If the decimal expansion of \( \pi \) contains arbitrarily long runs of 0s, then every \( n \) is included and therefore \( L = \emptyset \cup 1 \Sigma^* \); otherwise \( L \) is finite. Either way \( L \) is regular and hence decidable.

**d.** Undecidable. Let \( L \) denote the language in question. If \( L \) were decidable, then we would be able to check whether the language of any given Turing machine \( M \) is empty by running \( L \)'s decider on \( (M, q_{\text{accept}}) \). This would contradict Rice’s theorem, which asserts that every nontrivial property of the language of a Turing machine (such as emptiness) is undecidable.