1. True/false questions. Give a brief justification for your solution.
   
   a. In class we claimed that, if a polynomial-time algorithm is discovered for some problem, it is usually possible to discover a reasonably efficient algorithm, say one running in $O(n^5)$ time.
   
   We could try to formalize this by saying that $P \subseteq TIME(n^5)$. Is this a true statement?

   b. Another standard complexity class is $E = \bigcup_{c>0} TIME(2^{cn})$. Is it true that $E = EXP$?

2. In class we claimed that $P$ is a nice complexity class because polynomial-time computations are closed under composition. Let’s check whether polynomial-time computations are closed under a polynomial number of compositions.

   Let $M$ be a program with two inputs: $i \in \mathbb{N}$ and $w \in \Sigma^*$. Let $c > 0$ be a fixed constant (which depends on $M$ but not on $i$ or $w$) and let $n = |w|$. Suppose that

   - On any inputs, $M$ makes $\Theta(n^c)$ basic computational steps (each of which takes constant time).
   - $M(i, w)$ makes $\Theta(n^c)$ calls to $M(i+1, w)$ when $i < n$.
   - $M(n, w)$ only does basic computational steps, and does not calls $M$ again as a subroutine.

   Does $M(0, w)$ run in time polynomial in $n$?

3. a. Give an example of an infinite, decidable language $L$ (with, say, $\Sigma = \{0, 1\}$) satisfying the following property. For every Turing Machine $M$ that decides $L$, and every $x \in L$, the machine $M$ performs at least 100 steps before accepting $x$.

   b. Suppose there is a language $L$ and a TM $M$ such that, for every string $x$, $M$ halts on input $x$ after at most $\sqrt{|x|}$ steps. Prove that $M$ must actually halt on input $x$ after a constant number of steps.

   (Here $|x|$ denotes the length of string $x$. To avoid pedantic issues, let’s ignore the case $x = \epsilon$.)

   c. Let $t(n)$ be any increasing function that grows asymptotically slower than $n$, i.e., $t(n) \leq t(n+1)$ for all $n$, and $t(n) = o(n)$. (For example, $t(n) = \sqrt{n}$ or $t(n) = \log n$.) Prove that $\text{TIME}(t(n)) = \text{TIME}(1)$.

   (Pedantic detail: We assume that $t(n) \geq 1$ for all $n$.)

4. Let us consider a decision problem about a generalized form of Sudoku. (The case $n = 3$ corresponds to ordinary Sudoku.) A problem instance consists of an integer $n \geq 3$ and a two-dimensional grid of cells, with $n^2$ rows and $n^2$ columns. In the initial problem instance, each cell is either blank or contains a number in $\{1, \ldots, n^2\}$.

   The goal is to place a number into every blank cell such that:

   (1) Each column contain every number in $\{1, \ldots, n^2\}$ exactly once.
   (2) Each row contain every number in $\{1, \ldots, n^2\}$ exactly once.
(3) For every $i, j \in \{0, \ldots, n - 1\}$, the square at the intersection of rows \{ni + 1, \ldots, n(i + 1)\} and columns \{nj + 1, \ldots, n(j + 1)\} contains every number in \{1, \ldots, n^2\} exactly once.

[6] a. The decision problem $SUDOKU$ is: given an initial problem instance (in which each cell could be blank or contain a number), decide whether the blanks can be filled in such that conditions (1)-(3) are satisfied. Show that $SUDOKU$ is in NP.

[9] b. Suppose that someone proves that $SUDOKU$ (the decision problem) is in $P$. Give a polynomial-time algorithm with the following behavior: given an initial problem instance (in which each cell could be blank or contain a number), output either:

- A value for each cell such that conditions (1)-(3) are satisfied, or
- “Reject” if there is no way to satisfy conditions (1)-(3).

[2] 5. **OPTIONAL BONUS QUESTION:** This question relates to section 6.1 of the textbook, which discusses the Recursion Theorem (Theorem 6.3).

In class we used a reduction from $A_{TM}$ to prove that $REGULAR_{TM}$ is undecidable (see Theorem 5.3 in the text).

In this question, you must use the Recursion Theorem to prove to prove that $REGULAR_{TM}$ is undecidable.