FOUNDATIONS OF COMPUTING

Instructions: This exam consists of 7 problems, each of equal weight. Please write answers on the exam. If you would like to attach additional pages, include examinee number on each page. To keep answers concise, it is permissible to refer to algorithms in books and cite page numbers.
1. Consider three languages $A$, $B$, and $C$, where $A$ is known to be in $P$, $B$ is known to be in $NP$, and $C$ is known to be $NP$-complete. For each of the following statements, list the conclusions (if any) that can be drawn if the statement were proved. Your conclusions should be selected from the list (a)–(f) that follows the statements. For full credit, you need to list all conclusions that can be drawn from a statement (or write NONE if none applies). Note: $X \leq_P Y$ means $X$ reduces to $Y$ in polynomial time. Assume that $NP$-completeness is defined with respect to polynomial-time reductions (this is standard for most algorithms texts; some automata texts use log-space reductions).

   i. $C \leq_P A$
   ii. $B \leq_P C$
   iii. $C \leq_P B$
   iv. $A \leq_P B$

   v. Any algorithm that decides membership in $B$ requires at least $(1.0001)^n$ steps on a Turing machine, where $n$ is the input length.

Conclusions:

   a. $P = NP$
   b. $P \neq NP$
   c. $B$ is $NP$-complete
   d. $A$ is $NP$-complete
   e. $C$ is in $P$
   f. $B$ is in $P$
2. Of the three languages below, one is regular, one is context free but not regular, and another is not context free.

\[
L_1 = \{a^i b^j c^k \mid i = j + k\} \\
L_2 = \{a^i b^j c^k \mid i = \max(j, k)\} \\
L_3 = \{a^i b^j c^k \mid i \mod 2 = (j + k) \mod 2\}
\]

(a) Which language is which? [20%]

(b) Give a short, but convincing, argument for each of the four things that needs to be proved in order to establish the correct placement of \(L_1, L_2, L_3\) in the Chomsky hierarchy. There is no need to do correctness proofs of any grammars, regular expressions, or automata that you use in your arguments. [20% for each argument]
3. Let $T[1..n]$ be a sorted array of distinct integers. Give a fast divide-and-conquer algorithm that finds an index $i$ such that $T[i] = i$, if one exists. Give the recurrence which describes the time required by your algorithm. Find the solution to the recurrence ($\Theta$ class will suffice.)
4. For a Turing machine $T$, define $H(T) \subseteq \Sigma^*$ to be the set of words in $\Sigma^*$ on which $T$ halts. Prove that the following problem is undecidable: Given $T$, does $H(T)$ contain more than one word? Use the fact that the halting problem for Turing machines is undecidable.
5. Let \( S(n, k) \) be the number of ways to partition the set \( \{1, \ldots, n\} \) into \( k \) nonempty disjoint sets. For example, \( S(4, 2) = 7 \) since the ways to partition \( \{1, 2, 3, 4\} \) into 2 nonempty sets are:

\[
\begin{align*}
\{1, 2\} & \cup \{3, 4\} \\
\{1, 3\} & \cup \{2, 4\} \\
\{1, 4\} & \cup \{2, 3\}
\end{align*}
\]

\( S(n, k) \) is defined by the recurrence:

\[
S(n, k) = \begin{cases} 
1 & \text{if } n = k = 0 \\
0 & \text{if } n = 0 \text{ and } k > 0 \\
0 & \text{if } k = 0 \text{ and } n > 0 \\
S(n - 1, k - 1) + k \cdot S(n - 1, k) & \text{otherwise}
\end{cases}
\]

a. Write a dynamic programming algorithm to compute \( S(n, k) \).

b. Show the table of values of \( S(n, k) \) for all \( n \) and \( k \) satisfying \( 0 \leq n \leq 5 \) and \( 0 \leq k \leq 5 \). (Let the rows be indexed by \( n \) and the columns by \( k \).)
6. An AVL tree is a binary search tree in which at every node $x$, the heights of the left and right subtrees of $x$ differ by at most one.

a. What is the maximum number of nodes, $M(h)$ in an AVL tree of height $h$?

b. Find a recurrence relation which defines $m(h)$, the minimum number of nodes in an AVL tree of height $h$. (Don’t forget the basis.) You do not need to solve the recurrence.

c. Find a recurrence relation which defines $N(h)$, the number of AVL trees of height $h$. (Don’t forget the basis.) You do not need to solve the recurrence.
7. Design an algorithm that, given the adjacency list representation of an unweighted graph $G = (V, E)$ and a distinguished vertex $s \in V$, determines for each $v \in V$ the shortest path from $s$ to $v$. (The shortest path is the one with the minimum number of edges.) Your algorithm should run in time $O(n + m)$, where $n$ is the number of vertices and $m$ the number of edges in $G$. 