Problem 1. (20 points) Rank the following functions by order of growth; that is, find an arrangement $f_1, f_2, ..., f_n$ of the functions satisfying $f_1 = \Omega(f_2)$, $f_2 = \Omega(f_3)$, ..., $f_{n-1} = \Omega(f_n)$.

$$
\begin{array}{cccc}
n! & 4^{\lg n} & n \cdot 2^n & \sqrt{n} \\
e^n & 3^n & 2^n & n^2 \\
\lg^2 n & n^3 & 2^{\lg n} & n \log n \\
\log \log n & \lg n & n^{1/\lg n} & 2
\end{array}
$$

Problem 2. (20 points) The \textit{integer square root problem} is to determine the integer portion $p$ of the square root of integer $n$; that is, find $p = \lfloor \sqrt{n} \rfloor$.

(a) Give a linear-time algorithm to solve the integer square root problem. Prove your algorithm correct using a loop invariant.
(b) Give a logarithmic-time algorithm to solve the integer square root problem. Prove your algorithm correct using a loop invariant.

Problem 3. (20 points) Suppose that we have a hash table with $m$ slots.

(a) Describe two ways to resolve collisions.
(b) If collisions are resolved by chaining and $n$ keys are inserted into the table, assuming simple uniform hashing, what is the expected number of collisions?
(c) Under the same assumptions, what is the probability that exactly $k$ keys hash to a particular slot?

Problem 4. (20 points) Give an algorithm to determine whether a given node is a root of a valid binary search tree. Analyze the running time of your algorithm.

Problem 5. (20 points) The \textit{transpose} of a directed graph $G = (V, E)$ is the graph $G^T = (V, E^T)$, where $E^T = \{(u, v) \in V \times V : (v, u) \in E\}$. Thus, $G^T$ is $G$ with all its edges reversed. Give efficient algorithms for computing $G^T$ from $G$, for:

(a) Adjacency-list representation of $G$.
(b) Adjacency-matrix representation of $G$.

Analyze the running times of your algorithms.

Problem 6. (20 points) The \textit{knapsack problem} is the following. A thief robbing a store finds $n$ items. The $i$-th item is worth $v_i$ dollars and weighs $w_i$ pounds, where $v_i$ and $w_i$ are integers. The thief wants to take as valuable a load as possible, but they can carry at most $W$ pounds in their knapsack, for some integer $W$. Which items should they take?

Give a dynamic-programming solution to the knapsack problem.