Instructions: There are five problems in this exam. Some of them consist of several subproblems. Make sure you look through them carefully. Please write legibly.

1. Asymptotic Notation.

   a. Please rank the following functions by the order of their growth – from slowest to fastest: $4^{\log n}$, $n^2 \log n$, $(1.5)^n$, $\sqrt{4n}$, $2/n$.

   b. Please provide brief explanations to your answers for the following questions –

      (i) Is $\log n = O(n)$?
      (ii) Is $\log n = \Omega(n)$?
      (iii) Is $\log n = \Theta(n)$?
2. *Short questions.*

a. Compare and contrast separate chaining and linear probing for resolving collisions in hashing.

b. State the similarities and differences between an ordinary binary search tree and an AVL tree. Given the choice for implementing ordered dictionaries, when might you want to use one over the other?
3. *Binary Search Trees.* In the binary search tree below, assume that all the keys are distinct and that trees $T_0, T_1, \ldots, T_6$ are non-empty. That is, there are items stored in these trees. A node is identified as $x$. Please answer the following questions with explanations.

![Binary search tree diagram]

a. Describe the locations of the items with keys larger than $key[x]$.
b. Describe the location of $y$ such that $key[y]$ is the smallest key larger than $x$.
c. Describe the location of $z$ such that $key[z]$ is the largest key smaller than $x$.
d. Based on the observations you’ve made in answering the previous question, given a binary search tree $T$ whose keys are all distinct, describe an algorithm for implementing **removeGreaterThan**($k$) – a method that removes all items with keys larger than $k$. What is the runtime of your algorithm?
4. Splay Trees. Consider the splay tree $T$ below.

- Illustrate the changes in the splay tree when we access (or search) the nodes in order of their keys – from 1 to 6.
- Let us denote by $T'$ another splay tree different from $T$ that contained the keys 1 to 6. Suppose we also accessed the nodes in order of their keys (as in problem a). Do you expect to see the same result? That is, will the final tree be the same as that in a? Why or why not?
5. Suppose we are given an $n \times n$ array of distinct numbers such that (i) each individual row is increasing from left to right and (ii) each individual column is increasing from top to bottom. Design an algorithm that decides if a query number $x$ is in the array in $O(n)$-time.