Problem 0. Information (worth 1 point)

Write your name and KAIST ID at the top of **EACH** page on your answer paper.

Problem 1. (10 points)

You are given the following tree traversals of a binary search tree.

- Preorder: 6 2 1 4 3 5 8 7 12 10 9 11 15
- Inorder: 1 2 3 4 5 6 7 8 9 10 11 12 15

1) Draw a final resulting binary search tree that satisfies the given traversals above. You are NOT required to show ALL intermediate steps. (worth 7 points)
2) Explain how you find the root ***R*** of the entire tree, a left subtree rooted at the left child of the root ***R***, and a right subtree rooted at the right child of the root ***R***, based on the given tree traversals above. (worth 3 points)

Problem 2. (15 points)

Suppose that we store n elements in an m-size hash table ***T*** using separate chaining; we build a 2-3 tree for each of array positions instead of a linked list.

- Notes: Suppose that applying hash function takes constant time and m is equal to n. Accordingly, the load factor α = n/m = 1.

1) What is the average running time of insert and search in this hash table ***T*** under uniform hashing assumption?—Use a big-Oh notation (worth 4 points) Explain why (worth 4 points).
2) What is the worst-case running time of insert and search in this hash table ***T***?—Use a big-Oh notation (worth 4 points). Explain why (worth 3 points).

Problem 3. (14 points)

1) Let an array pq[ ] be a max-oriented binary heap that contains the N distinct integers 1, 2, …, N in pq[1] through pq[N]. Then, key N must be in pq[1]; key N-1 must be in either pq[2] or pq[3]; and key N-2 must be in either pq[2] or pq[3].
   A. Is the statement above true or false? (worth 1 point)
   B. Justify your answer. (worth 6 points)

2) In a binary search tree, we can find the predecessor (which is the next smallest element) of a given key in O(1) time. Assume that all keys stored in a tree are distinct.
   A. Is the statement above true or false? (worth 1 point)
   B. Justify your answer. (worth 6 points)

Problem 4. (10 points)

Draw a resulting AVL tree after inserting the keys: 25, 35, 45, 20, 10 in that order into an initially empty AVL tree. Show ALL intermediate steps—draw a tree after **EACH** key insertion and after **EACH** tree rotation.