TMA 16 – Assignment (MAMN) 16

Relevant Study Material: Chapter 12, and the section in the study guide on **AVL trees**
Weight: 3 points
Number of Questions: 5
Semester: 2025B
Submission Deadline: May 28, 2025

Question 1 (20 points; 5 for a, 8 for b, 7 for c)

The **preorder** and **inorder** traversal lists of a binary tree are given below (read from left to right):

- **Preorder traversal**: D, E, B, C, A, F, I, G, H
- **Inorder traversal**: B, E, C, D, I, F, G, A, H

a. Draw a diagram of the **original binary tree**.

b. In general, explain how a binary tree can be reconstructed from its **preorder** and **inorder** traversal sequences.
Describe the reconstruction process as clearly and concisely as possible.

c. Analyze the **time- complexity** of the reconstruction algorithm in both the **best case** and the **worst case**.

Question 2 (20 points; 10 points per sub-question)

a. Write an algorithm that receives a pointer to a **binary tree** and checks whether it is a **binary search tree (BST)**.

b. Analyze the **time and space complexity** of the algorithm you wrote.

Question 3 (15 points; 7 for a, 8 for b)

Draw a **balanced binary tree** containing two **leaf nodes** x and y, such that:

$$\left| level\,(x) - level\,(y) \right| = 2$$

That is, the difference in levels between x and y is exactly 2.

- Let n represent the total number of nodes in the tree you construct.
- Insert the values 1, 2.... n into the tree in such a way that it becomes a **binary search tree**.

Question 4 (20 points; 11 for a, 9 for b)

a. Prove or disprove (by counterexample) the following claim:
If T is an **AVL tree**, then there do not exist two leaf nodes in T whose level difference is greater than 2.

Note: The level of a node in a tree is defined as its depth (as in Section 5.2 of the textbook).

b. Suppose a new element is inserted into a (not necessarily balanced) **binary search tree**, and then immediately deleted.
Will the resulting tree always be identical to the original?

Provide a well-reasoned explanation.

Question 5 (25 points)

Let k=⟨k1,k2⟩ be a **composite key**, consisting of two parts k1 and k2, both of which are real numbers.

Design a data structure that allows efficient implementation of the following operations on a set SS of elements with keys of this type:

- **INSERT (S, k)**: Insert an element with key k into S, but only if no element with the same key already exists.
- **DELETE (S, k)**: Delete the element in S with key k.
- **DELETE-ALL (S, k_1)**: Delete **all elements** from S whose first component k_1 matches the given value.

For each operation:

- Explain how it will be executed.
- Analyze the **runtime complexity** of the operation.
