EC-200 Data Structures

LAB MANUAL # 05

**Course Instructor:** Dr. Anum Abdul Salam

**Lab Engineer:** Engineer Hira Irshad

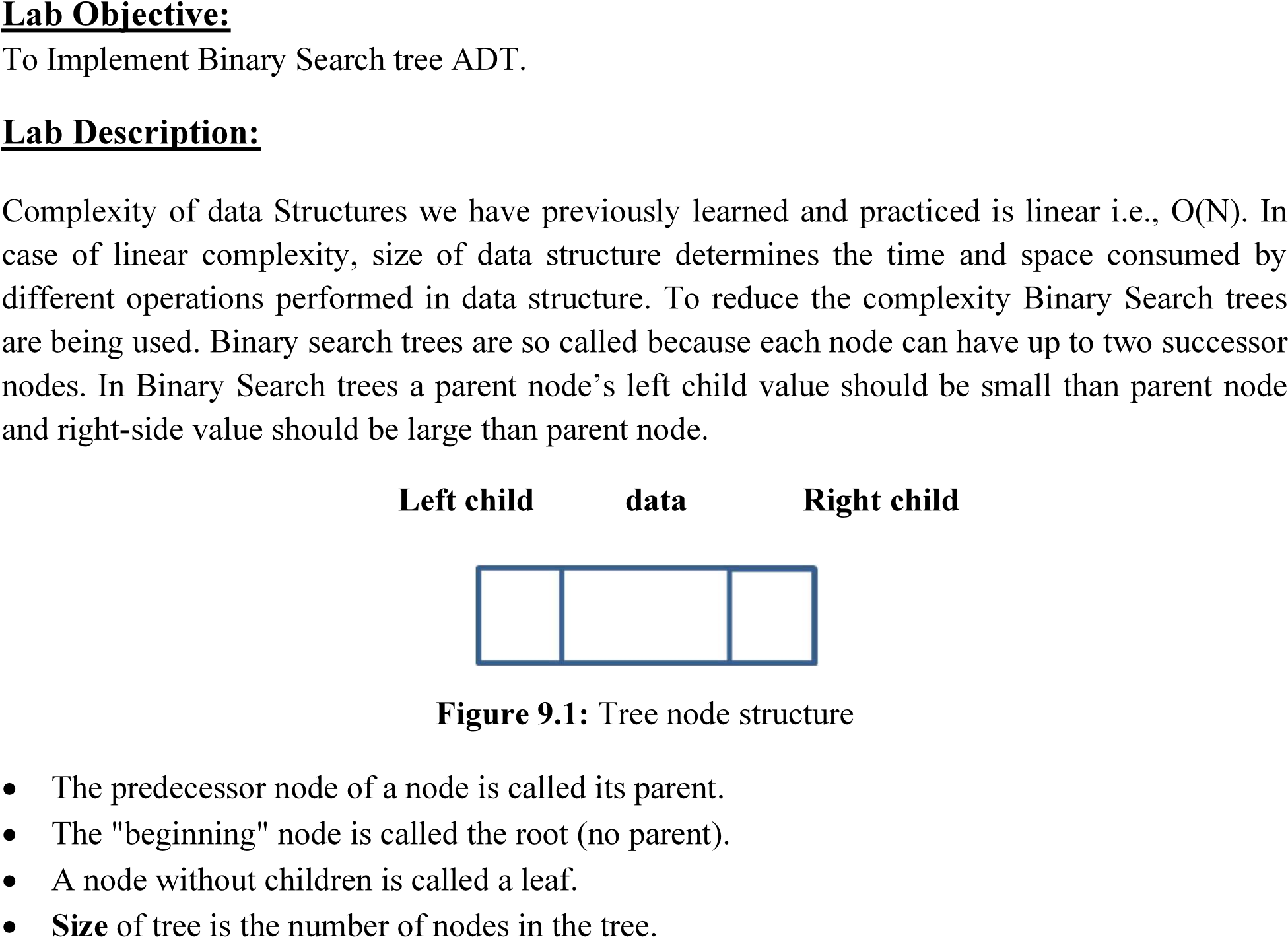
**Student Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

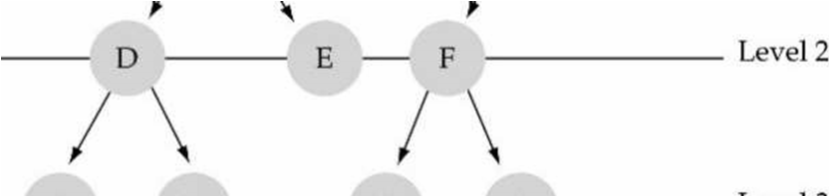
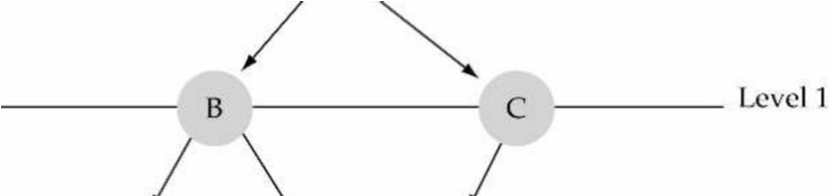
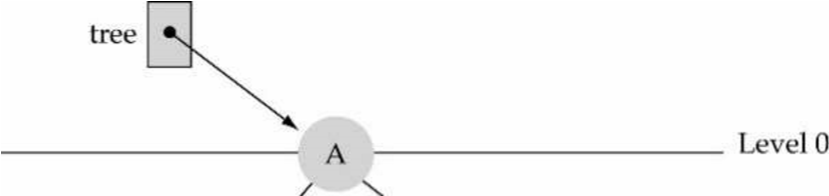
**Degree/ Syndicate: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

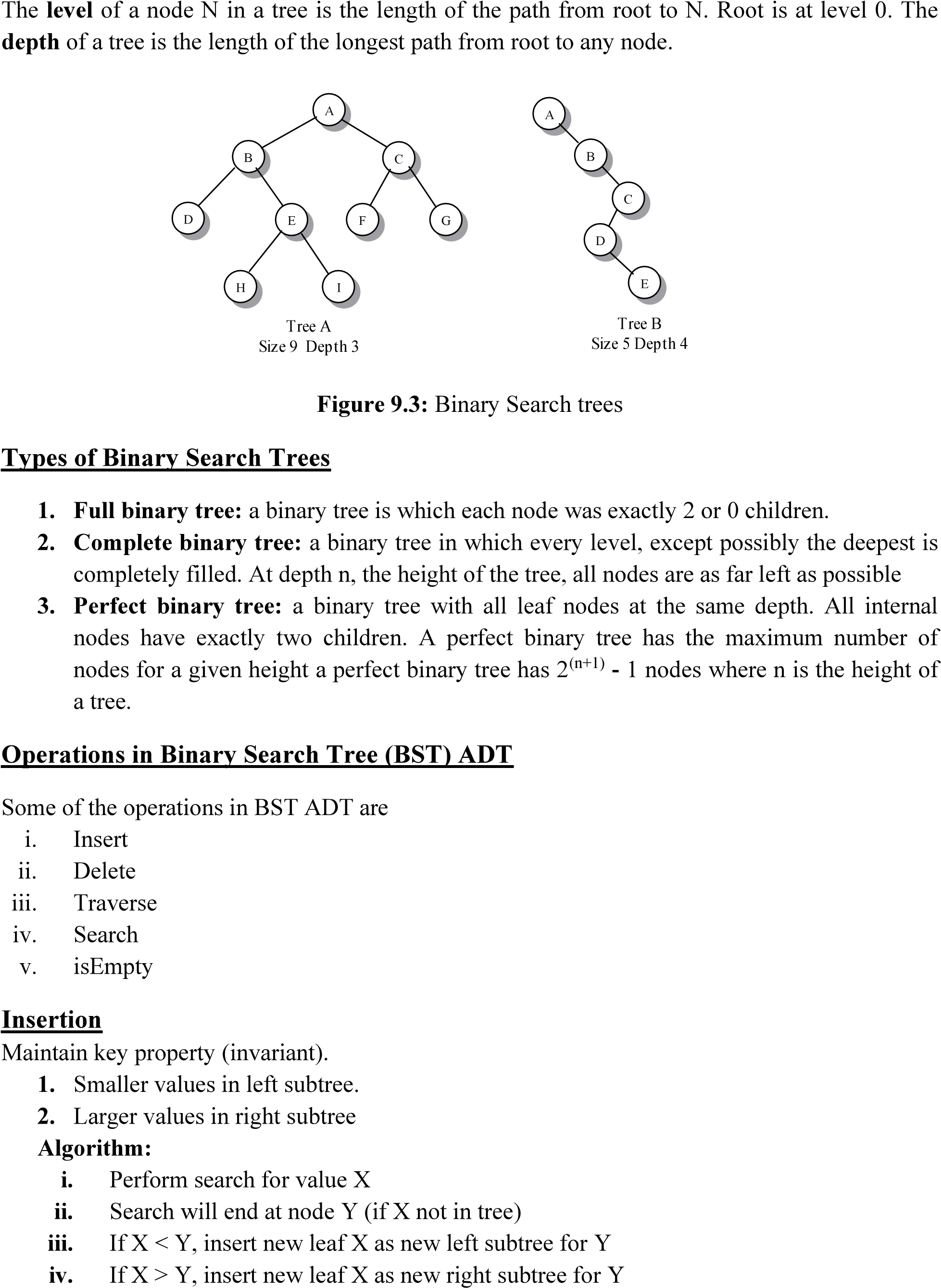
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Trait** | **Obtained Marks** | **Maximum Marks** |
| **R1** | **Application Functionality 20%** |  | 20 |
| **R2** | **Specification & Data structure implementation**  **30%** |  | 30 |
| **R3** | **Reusability**  **10%** |  | 10 |
| **R4** | **Input Validation**  **10%** |  | 10 |
| **R5** | **Efficiency**  **20%** |  | 20 |
| **R6** | **Delivery**  **10%** |  | 10 |
| **R7** | **Plagiarism above 60%** |  | 0 |
|  | **Total** |  | 100 |

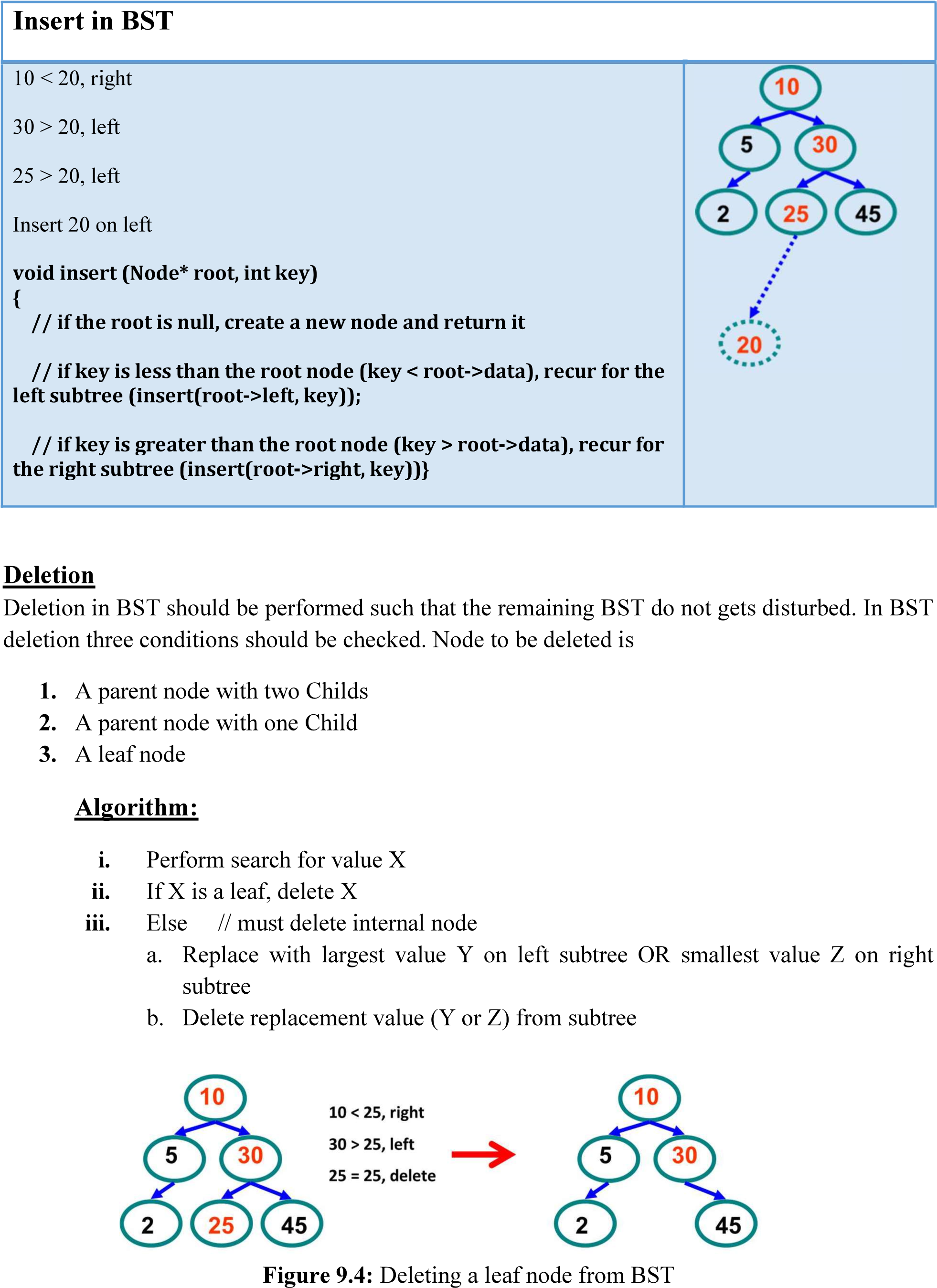
**Total Marks = O**𝒃𝒕𝒂𝒊𝒏𝒆𝒅 𝑴𝒂𝒓𝒌𝒔 (∑6𝟏 𝑹𝒊 ∗ 𝑹7)

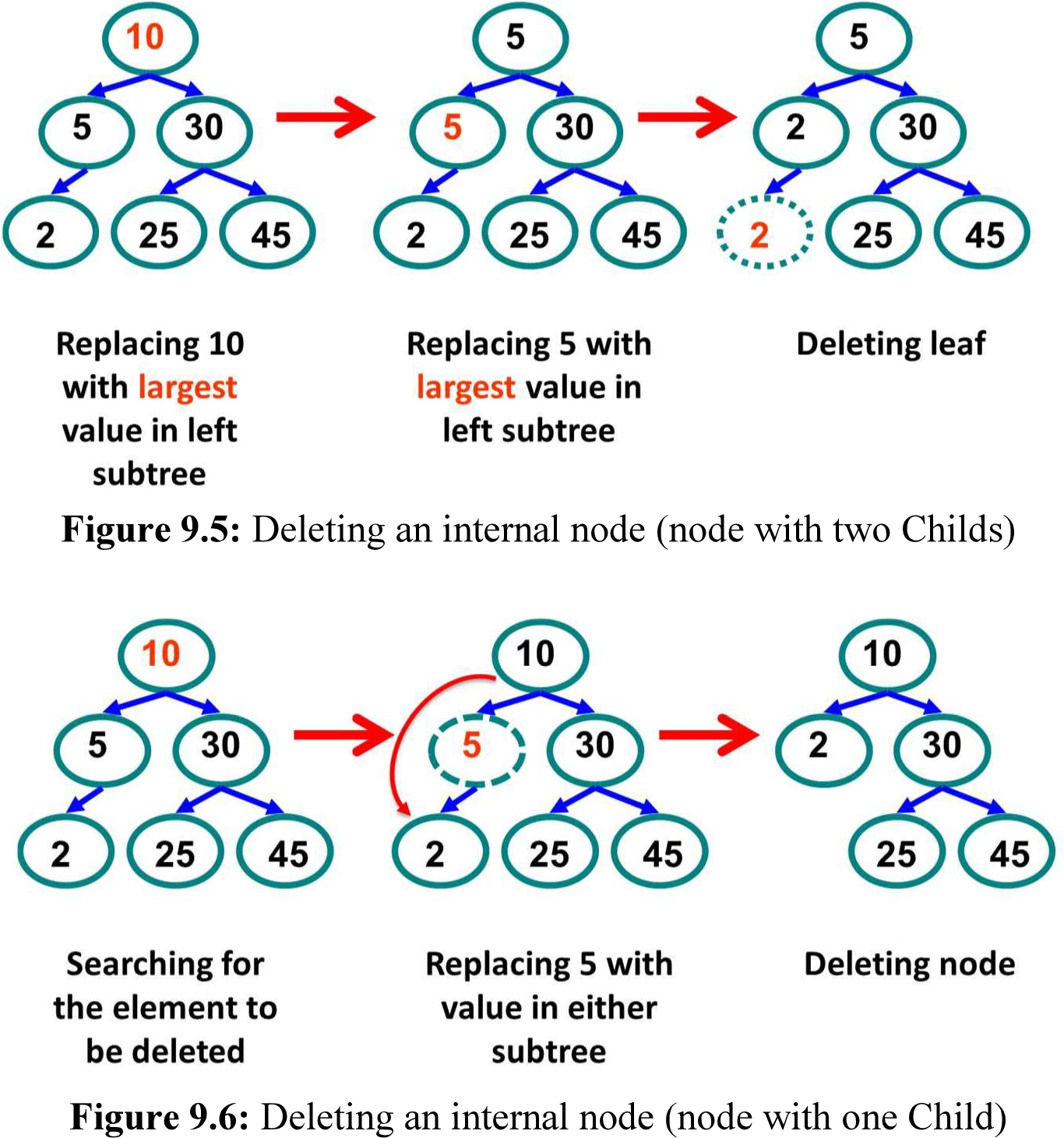
**Lab 05: Binary Search Tree**

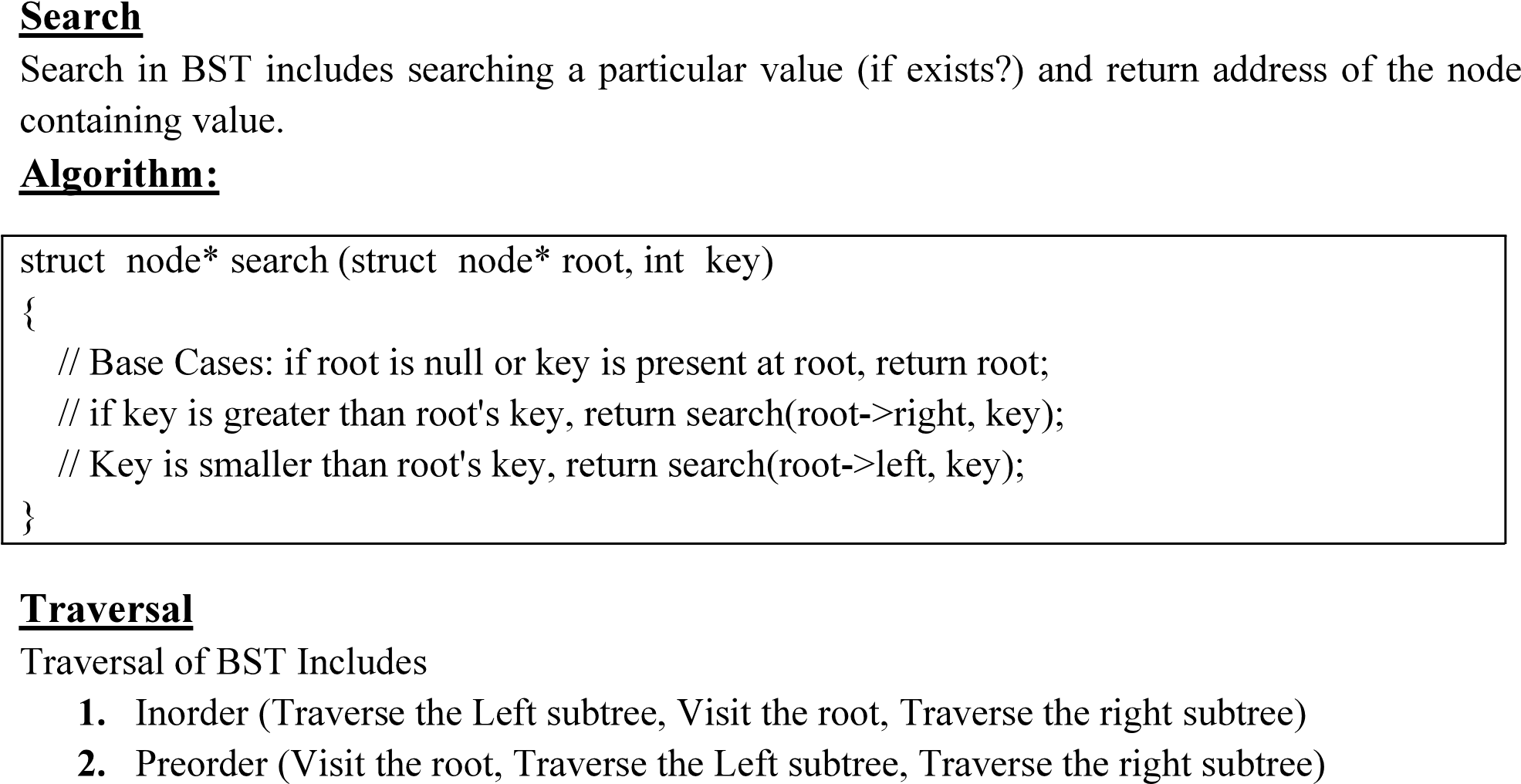


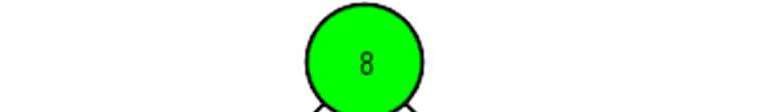












**LAB TASKS**

***Note: Implement the following tasks using templates.***

**1. Add these functions to the Binary Search Tree ADT from pre-lab task: deletion, traversing (inorder, preorder, postorder), copy constructor and destructor.**

**TEST PLAN:**

Execute your test plan. If you discover mistakes in your implementation of the dynamic tree ADT, correct them and execute your test plan again.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

2. A library wants to manage its book catalog using a **popularity-based ranking system**. Each book has a popularity score and unique identifier. The system must:

* Maintain the catalog **sorted by popularity** for easy ranking.
* Support **dynamic insertion and deletion** of books.
* Support **searching, top-k queries, and range queries** based on popularity.

Each book record contains:

* Book ID (integer)
* Title (string)
* Author (string)
* Popularity score (integer)

The BST is **ordered by popularity score**.

Students must implement the following BST operations:

* **Insert(Book data)** – Add a new book to the BST in correct order.
* **Delete(int popularity)** – Remove a book while maintaining BST structure.
* **Search(int popularity)** – Locate a specific book and return its details.
* **Display()** – Display all books sorted by popularity.
* **CountNodes()** – Count total books in the BST.
* **TopKBooks(int k)** – Display the top k books with the highest popularity scores.
* **RangeQuery(int minScore, int maxScore)** – List all books with popularity scores within a given range.

