**DS lab 08 Exercise Tasks**

**Question#1:**

Implement a binary search tree and perform all operations you learned above like: Inserting, Deleting, Searching, and Traversing.

**Question#2:** Reuse the methods from above **question 1** and complete below question:

You are required to implement a game titled **“BattleQuest: BST Arena”** using a **Binary Search Tree (BST)** where each node in the tree stores a **Combatant object**. This game simulates a classic RPG-style battle between a player’s team of heroes and a team of enemies. The core challenge involves managing two separate **BSTs** — one for the player’s team and another for the enemy team — where each node contains a **Combatant object** with attributes such as name, health points (HP), and attack power.

The game follows a strict turn-based combat system, where the player’s team and the enemy team alternate turns. Only the frontline combatant — defined as the node with the smallest key value (i.e., the leftmost node in the BST) participates in combat during each turn.

When a combatant’s health drops to zero or below, that combatant is removed from their respective BST. The next available combatant in the tree (the new leftmost node) becomes the frontline fighter for the following turn.

The battle continues until one team’s BST becomes empty, determining the winner.

The player’s team should be initialized with five heroes. The enemy team should be initialized with five enemies.

Both trees should be constructed using insertions according to the BST property based on the combatant’s name or unique ID.

The player always attacks first in each round, followed by the enemy counterattack. During an attack: The attacker’s damage = base attack power + random value (0 to 4)

The damage will affect the defender’s health. If the defender’s health ≤ 0, they are removed from their BST immediately. The next combatant automatically takes their place as the new frontline fighter.

During each turn, the game must display the following information clearly:

* Round number
* Names and current HP of all heroes and enemies (in BST order)
* Active attackers and defenders
* Damage dealt in the attack
* Updated HP values after each hit
* Elimination messages when a combatant is defeated
* Victory message when one BST becomes empty

**Question#3**

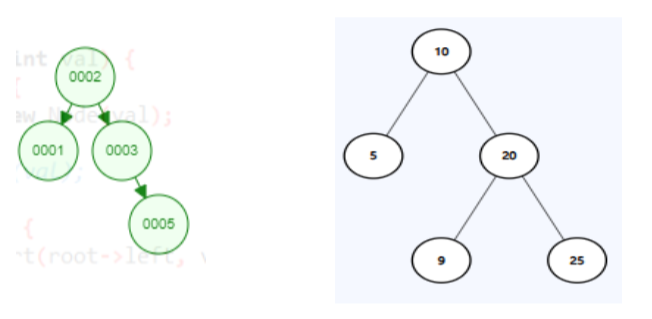
Create the BST which stores the students object in BST and student object has the property like name, roll and score. Create tree based on name of student AND Perform following operations in BST:

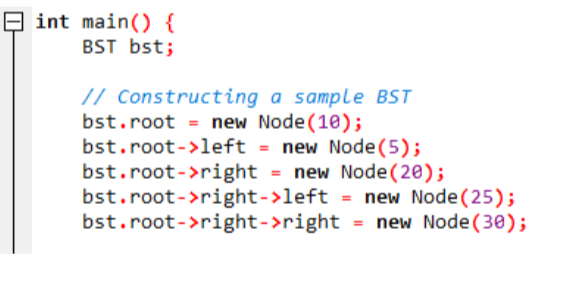
* Insert the student record,
* Search the students based on name
* Delete the students whose score is less than 10
* And get the students whose score is max in tree.

In main function create the object array of students with 10 students from those store 7 seven students randomly and perform all above objects.

**Question#4:** Search for the value defined by the user in the tree. If the value does not exist insert it and print the new tree.

**Question#5:** Given the root of a binary tree. Check whether these are Binary Search Tree or not.





**Question#6:** Suppose you are working on a project for a small online retailer that needs to keep track of their inventory using a binary search tree. The retailer’s inventory consists of a unique product ID and its corresponding quantity in stock. Write a C++ class for the binary search tree and add the required functionalities to insert new products into the tree, update the quantity of existing products, and search for products by their ID.

Additionally, the retailer would like to keep track of the product with the highest quantity in

stock. Implement a function that returns the ID of this product, along with its quantity.

**Question#7:** Given a Binary Search Tree, find the median of it.

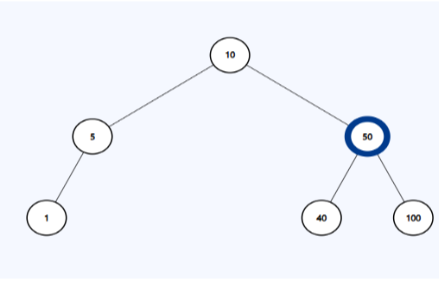
* If number of nodes are even: then median = ((n/2 + ((n)/2+1) ) /2
* If number of nodes are odd: then median = (n+1)/2

**Question#7:**

Given a Binary Search Tree (BST) and a range [a, b], the task is to count the number of nodes in the BST that lie in the given range.

Examples:

Input: a= 5, b = 45



Output: 3

Explanation: There are three nodes in range [5, 45] = 5, 10 and 40.