Binary Search Tree (BST) – Coding Problem Set (Student Version)

Problem 1: Insert into BST Write a method to insert a value into a BST. Method: `public Node insert(Node root, int key)`

Problem 2: Search in BST Implement a recursive method that searches for a key in a BST. Method: `public boolean search(Node root, int key)`

Problem 3: Delete from BST Write a method to delete a node from a BST, handling nodes with 0, 1, or 2 children. Method: `public Node delete(Node root, int key)`

Problem 4: Inorder Traversal Implement a method to perform inorder traversal of a BST. Method: `public void inorder(Node root)`

Problem 5: Preorder Traversal Implement a method to perform preorder traversal of a BST. Method: `public void preorder(Node root)`

Problem 6: Postorder Traversal Implement a method to perform postorder traversal of a BST. Method: `public void postorder(Node root)`

Problem 7: Find Minimum Write a method to find the minimum value in a BST. Method: `public int findMin(Node root)`

Problem 8: Find Maximum Write a method to find the maximum value in a BST. Method: `public int findMax(Node root)`

Problem 9: Validate BST Write a method to check whether a binary tree is a valid BST. Method: `public boolean isBST(Node root, int min, int max)`

Problem 10: Lowest Common Ancestor Write a method to find the lowest common ancestor of two nodes in a BST. Method: `public Node lowestCommonAncestor(Node root, int p, int q)`

Problem 11: K-th Smallest Element Write a method to find the k-th smallest element in a BST. Method: `public int kthSmallest(Node root, int k)`

Problem 12: Height of BST Write a method to compute the height of a BST. Method: `public int height(Node root)`

Problem 13: Range Sum Write a method to calculate the sum of all node values within a given range [L, R]. Method: `public int rangeSumBST(Node root, int L, int R)`

Problem 14: Inorder Successor Write a method to find the inorder successor of a given node in a BST. Method: `public Node inorderSuccessor(Node root, Node target)`

Problem 15: Inorder Predecessor Write a method to find the inorder predecessor of a given node in a BST. Method: `public Node inorderPredecessor(Node root, Node target)`

Problem 16: Check Balanced BST Write a method to check if a BST is height-balanced. Method: `public boolean isBalanced(Node root)`

Problem 17: Sorted Array to BST Write a method to convert a sorted array into a height-balanced BST. Method: `public Node sortedArrayToBST(int[] nums, int start, int end)`

Problem 18: BST to Linked List Write a method to flatten a BST into a sorted linked list (inorder order). Method: `public Node bstToLinkedList(Node root)`

Problem 19: Path Sum Write a method to check if a BST has a root-to-leaf path with a given sum. Method: `public boolean hasPathSum(Node root, int targetSum)`

Problem 20: Range Count Write a method to count how many nodes have values within a given range [L, R]. Method: `public int rangeCountBST(Node root, int L, int R)`

Problem 21: Mirror BST Write a method to convert a BST into its mirror image. Method: `public Node mirrorBST(Node root)`

Problem 22: BST to Greater Sum Tree Write a method to transform a BST so that each node contains the sum of all greater or equal nodes. Method: `public Node bstToGreaterSumTree(Node root)`

Problem 23: Serialize BST Write a method to serialize a BST into a string. Method: `public String serialize(Node root)`

Problem 24: Deserialize BST Write a method to deserialize a string back into a BST. Method: `public Node deserialize(String data)`