**CSE220: Data Structures (Lab)**

**Fall 2024**

**Lab Quiz - 05 [Solution]**

### **Question 1 [7][A]**

**Rubric:**

* Base case handling: 1
* Single child check: 2
* Level check: 1
* Appropriate sum and recursive call: 3

**Python implementation:**

def sum\_odd\_single\_child(root, level = 0):

if root is None: return 0

sum = 0

if (root.left and not root.right) or (root.right and not root.left):

if level % 2 == 1: sum = root.val

return sum + sum\_odd\_single\_child(root.left, level + 1) + sum\_odd\_single\_child(root.right, level + 1)

**JAVA implementation:**

public static int sumOddSingleChild(Node root, int level) {

if (root == null) return 0;

int sum = 0;

if ((root.left != null && root.right == null) || (root.right != null && root.left == null)) {

if (level % 2 == 1) sum = root.val;

}

return sum + sumOddSingleChild(root.left, level + 1) + sumOddSingleChild(root.right, level + 1);

}

**Note:** For set [B], just change the level checking logic.

**Question 2 [8]**

**Rubric:**

* Base case check: 1
* Handling exact match of x to a node in BST: 1
* Searching in left subtree [set A] or right subtree [set B]: 1.5
* Searching in the other subtree: 1.5
* Validating the floor/ceil obtained from the other subtree before returning: 2
* Appropriate return value: 1

**Python implementation:**

| **Set: A** | **Set: B** |
| --- | --- |
| def get\_floor\_value(root, x):  if not root: return -1  if root.val == x: return root.val  if root.val > x:  return get\_floor\_value(root.left, x)  floor = get\_floor\_value(root.right, x)  if floor != -1 and floor <= x:  return floor  return root.val | def get\_ceil\_value(root, x):  if not root: return -1  if root.val == x: return root.val  if root.val < x:  return get\_ceil\_value(root.right, x)  ceil = get\_ceil\_value(root.left, x)  if ceil != -1 and ceil >= x:  return ceil  return root.val |

**Java implementation:**

| **Set: A** | **Set: B** |
| --- | --- |
| public static int getFloorValue(Node root, int x) {  if (root == null) return -1;  if (root.val == x) return root.val;  if (root.val > x)  return getFloorValue(root.left, x);  int floor = getFloorValue(root.right, x);  return (floor != -1 && floor <= x) ? floor : root.val;  } | public static int getCeilValue(Node root, int x) {  if (root == null) return -1;  if (root.val == x) return root.val;  if (root.val < x)  return getFloorValue(root.right, x);  int ceil = getCeilValue(root.left, x);  return (ceil != -1 && ceil >= x) ? ceil : root.val;  } |