**l.RUBRIC**

| **Category** | **Marks** |
| --- | --- |
| **Proper method/function definition with proper parameters** | **3** |
| **Properly Calculating the summation of all odd number in left subtree** | **5** |
| **Properly Calculating the summation of all even number in right subtree** | **5** |
| **return the result** | **2** |
| **Note: There can be multiple solutions which produce the correct answer. Use your best judgement to evaluate the scripts.** | |

**SET- A**

| Python | Java |
| --- | --- |
| def sum\_odd(node):  if not node:  return 0  if node.elem % 2 != 0:  val = node.elem  else:  val=0  return val + sum\_odd(node.left) + sum\_odd(node.right)  def sum\_even(node):  if not node:  return 0  if node.elem % 2 == 0:  val = node.elem  else:  val = 0  return val + sum\_even(node.left) + sum\_even(node.right)  def subtree\_difference(root):  left\_sum = sum\_odd(root.left)  right\_sum = sum\_even(root.right)    return abs(left\_sum - right\_sum) | **public static int sum\_odd(Node node) {**  **if (node == null) {**  **return 0;**  **}**  **if (node.elem % 2 != 0){**  **int val = node.elem;**  **}**  **else{ int val= 0;}**  **return val \* sum\_odd(node.left) \* sum\_odd(node.right);**  **}**  **public static int sum\_even(Node node) {**  **if (node == null) {**  **return 1;**  **}**  **if (node.elem % 2 == 0){**  **int val = node.elem;**  **}**  **else{ int val= 1;}**  **return val \* sum\_even(node.left) \* sum\_even(node.right);**  **}**  **public static int subtree\_difference(Node root) {**  **if (root == null) {**  **return 0;**  **}**  **int leftSum = sum\_odd(root.right);**  **int rightSum= sum\_even(root.left); return Math.abs(leftSum - rightSum);**  **}** |

**SET-B**

| **Python** | **JAVA** |
| --- | --- |
| **def mul\_odd(node):**  **if not node:**  **return 1**  **if node.elem % 2 != 0:**  **val = node.elem**  **else:**  **val=1**  **return val \* mul\_odd(node.left) \* mul\_odd(node.right)**  **def mul\_even(node):**  **if not node:**  **return 1**  **if node.elem % 2 == 0:**  **val = node.elem**  **else:**  **val= 1**  **return val \* mul\_even(node.left) \* mul\_even(node.right)**  **def subtree\_difference(root):**  **left\_mul = mul\_odd(root.right)**  **right\_mul = mul\_even(root.left)**    **return abs(left\_mul - right\_mul)** | **public static int mul\_odd(Node node) {**  **if (node == null) {**  **return 1;**  **}**  **if (node.elem % 2 != 0){**  **int val = node.elem;**  **}**  **else{ int val= 1;}**  **return val \* mul\_odd(node.left) \* mul\_odd(node.right);**  **}**  **public static int mul\_even(Node node) {**  **if (node == null) {**  **return 1;**  **}**  **if (node.elem % 2 == 0){**  **int val = node.elem;**  **}**  **else{ int val= 1;}**  **return val \* mul\_even(node.left) \* mul\_even(node.right);**  **}**  **public static int subtreeDifference(Node root) {**  **if (root == null) {**  **return 0;**  **}**  **int leftProduct = mul\_odd(root.right);**  **int rightProduct = mul\_even(root.left); return Math.abs(leftProduct - rightProduct);**  **}** |