Solve

## **SET#A**

| **PYTHON** | **JAVA** |
| --- | --- |
| def find\_mirror(left, right, target):  if left is None or right is None:  return None  if left.data == target:  return right.data  if right.data == target:  return left.data  res = find\_mirror(left.left, right.right, target)  if res is not None:  return res  return find\_mirror(left.right, right.left, target)  def mirror\_parity(root, x):  if root is None:  return "No Mirror Found"  mirror\_val = find\_mirror(root.left, root.right, x)  if mirror\_val is None:  return "No Mirror Found"  return "Even" if mirror\_val % 2 == 0 else "Odd" | public class BSTMirrorOperations {  private static Integer findMirror(Node left, Node right, int target) {  if (left == null || right == null)  return null;  if (left.data == target)  return right.data;  if (right.data == target)  return left.data;  Integer res = findMirror(left.left, right.right, target);  if (res != null)  return res;  return findMirror(left.right, right.left, target);  }  public static String mirrorParity(Node root, int x) {  if (root == null)  return "No Mirror Found";  Integer mirrorVal = findMirror(root.left, root.right, x);  if (mirrorVal == null)  return "No Mirror Found";  return (mirrorVal % 2 == 0) ? "Even" : "Odd";  }  } |

## **RUBRIC**

| **Category** | **Marks** |
| --- | --- |
| Correctly identifying the mirror node using recursion and BST properties. | 3+3 |
| Code successfully gives the correct output for all types of cases | 3+2 |
| Properly handles cases like empty tree, missing node, or no mirror node. | 2+2 |

## 

## 

## **SET#B**

| **PYTHON** | **JAVA** |
| --- | --- |
| def find\_mirror(left, right, target):  if left is None or right is None:  return None  if left.data == target:  return right.data  if right.data == target:  return left.data  res = find\_mirror(left.left, right.right, target)  if res is not None:  return res  return find\_mirror(left.right, right.left, target)  def mirror\_multiply(root, x):  if root is None:  return "No Mirror Found"  mirror\_val = find\_mirror(root.left, root.right, x)  if mirror\_val is None:  return "No Mirror Found"  return x \* mirror\_val | public class BSTMirrorOperations {  private static Integer findMirror(Node left, Node right, int target) {  if (left == null || right == null)  return null;  if (left.data == target)  return right.data;  if (right.data == target)  return left.data;  Integer res = findMirror(left.left, right.right, target);  if (res != null)  return res;  return findMirror(left.right, right.left, target);  }  public static Object mirrorMultiply(Node root, int x) {  if (root == null)  return "No Mirror Found";  Integer mirrorVal = findMirror(root.left, root.right, x);  if (mirrorVal == null)  return "No Mirror Found";  return x \* mirrorVal;  } |

## **RUBRIC**

| **Category** | **Marks** |
| --- | --- |
| Correctly identifying the mirror node using recursion and BST properties. | 3+3 |
| Code successfully gives the correct output for all types of cases | 3+2 |
| Properly handles cases like empty tree, missing node, or no mirror node. | 2+2 |