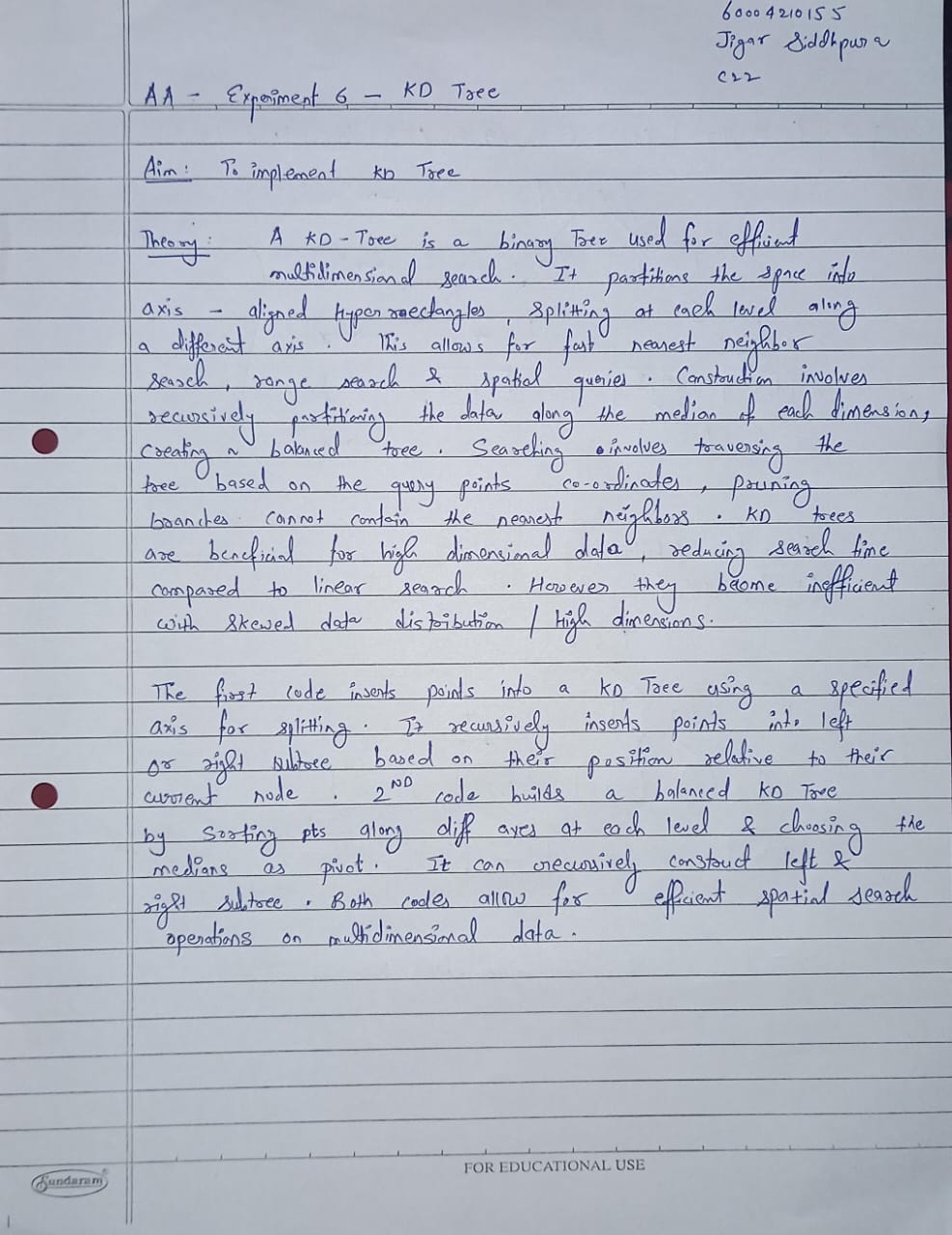
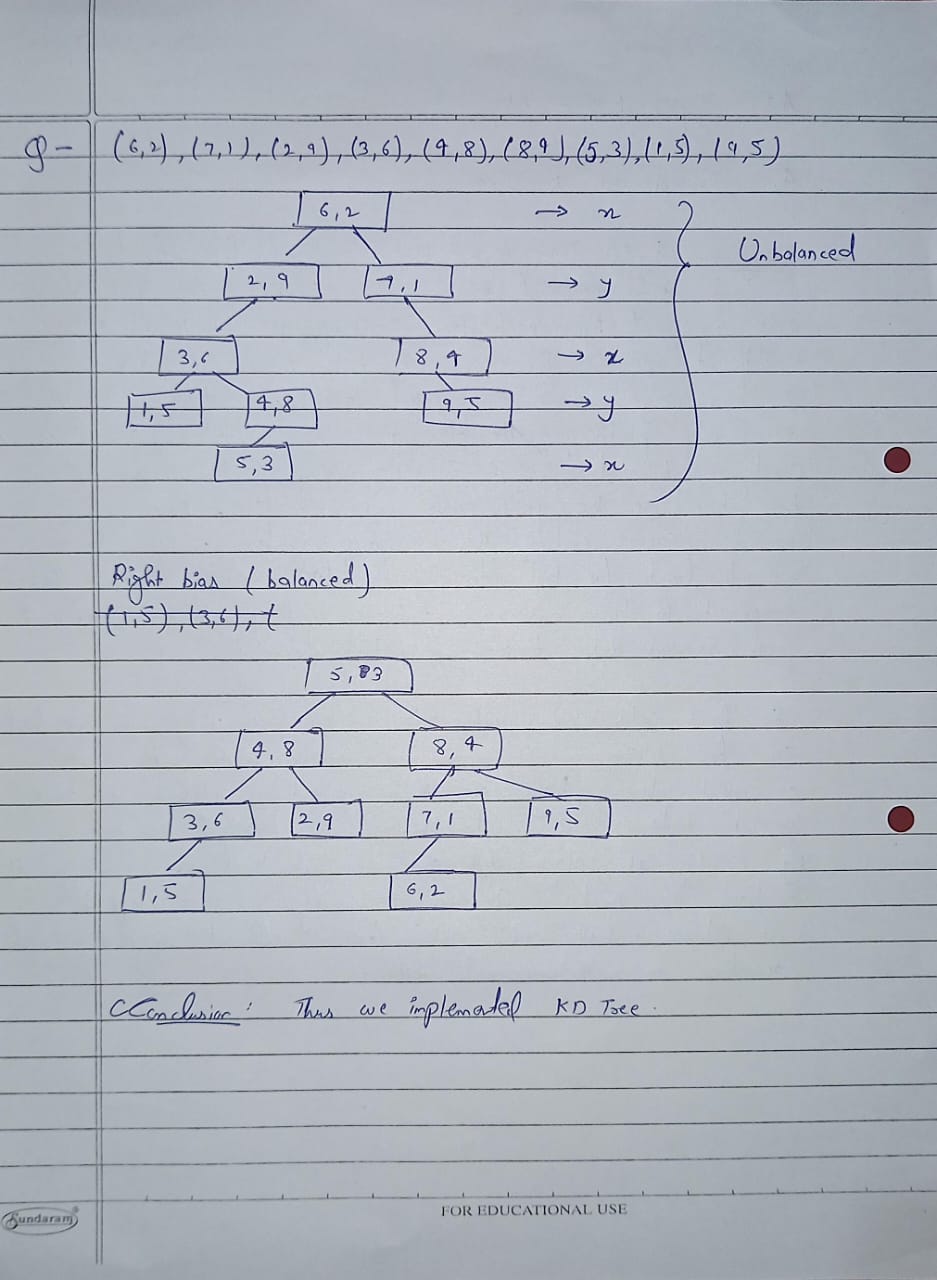
**Name:** Jigar Siddhpura **SAPID:** 60004200155

**DIV:** C/C2 **Branch:** Computer Engineering

AA - Experiment 5 - KD Tree





**CODE :**

**UNBALANCED KD TREE:**

class Node:

    def \_\_init\_\_(self, point, axis):

        self.point = point

        self.axis = axis

        self.left = None

        self.right = None

def insert(root, point, axis=0):

    if root is None:

        return Node(point, axis)

    if point[axis] < root.point[axis]:

        root.left = insert(root.left, point, (axis + 1) % len(point))

    else:

        root.right = insert(root.right, point, (axis + 1) % len(point))

    return root

def print\_tree(node, level=0, side=None):

    if node is not None:

        prefix = ""

        if side is not None:

            prefix = side + "---- "

        print("   " \* level + prefix + str(node.point))

        print\_tree(node.left, level + 1, "L")

        print\_tree(node.right, level + 1, "R")

# Sample data points

points = [[6, 2], [7, 1], [2, 9], [3, 6], [4, 8], [8, 4], [5, 3], [1, 5], [9, 5]]

# points = [[6, 2, 9], [7, 1, 2], [2, 9, 6], [3, 6, 1], [4, 8, 5], [8, 4, 4], [5, 3, 7], [1, 5, 1], [9, 5, 4]]

# Build KDTree

root = None

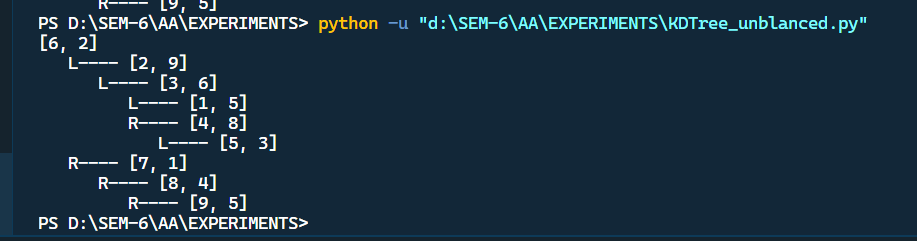
for point in points:

    root = insert(root, point)

# Print the tree

print\_tree(root)

**OUTPUT :**



**BALANCED KD TREE ( Right biased ):**

class Node:

    def \_\_init\_\_(self, point, axis):

        self.point = point

        self.axis = axis

        self.left = None

        self.right = None

def build\_kdtree(points, depth=0):

    if not points:

        return None

    # Select axis based on depth so that axis cycles through all valid values

    k = len(points[0])  # Dimension of the points

    axis = depth % k

    # Sort points based on the axis and choose median as pivot element

    points.sort(key=lambda x: x[axis])

    median = len(points) // 2

    # Create node and construct subtrees

    node = Node(points[median], axis)

    node.left = build\_kdtree(points[:median], depth + 1)

    node.right = build\_kdtree(points[median + 1:], depth + 1)

    return node

def print\_tree(node, level=0, side=None):

    if node is not None:

        prefix = ""

        if side is not None:

            prefix = side + "---- "

        print("   " \* level + prefix + str(node.point))

        print\_tree(node.left, level + 1, "L")

        print\_tree(node.right, level + 1, "R")

# Sample data points

points = [[6, 2], [7, 1], [2, 9], [3, 6], [4, 8], [8, 4], [5, 8], [1, 5], [9, 5]]

# points = [[6, 2, 9], [7, 1, 2], [2, 9, 6], [3, 6, 1], [4, 8, 5], [8, 4, 4], [5, 3, 7], [1, 5, 1], [9, 5, 4]]

# Build balanced KDTree

root = build\_kdtree(points)

# Print the tree

print\_tree(root)

**OUTPUT :**

