ADVANCED DATA STRUCTURES and ALGORITHMS

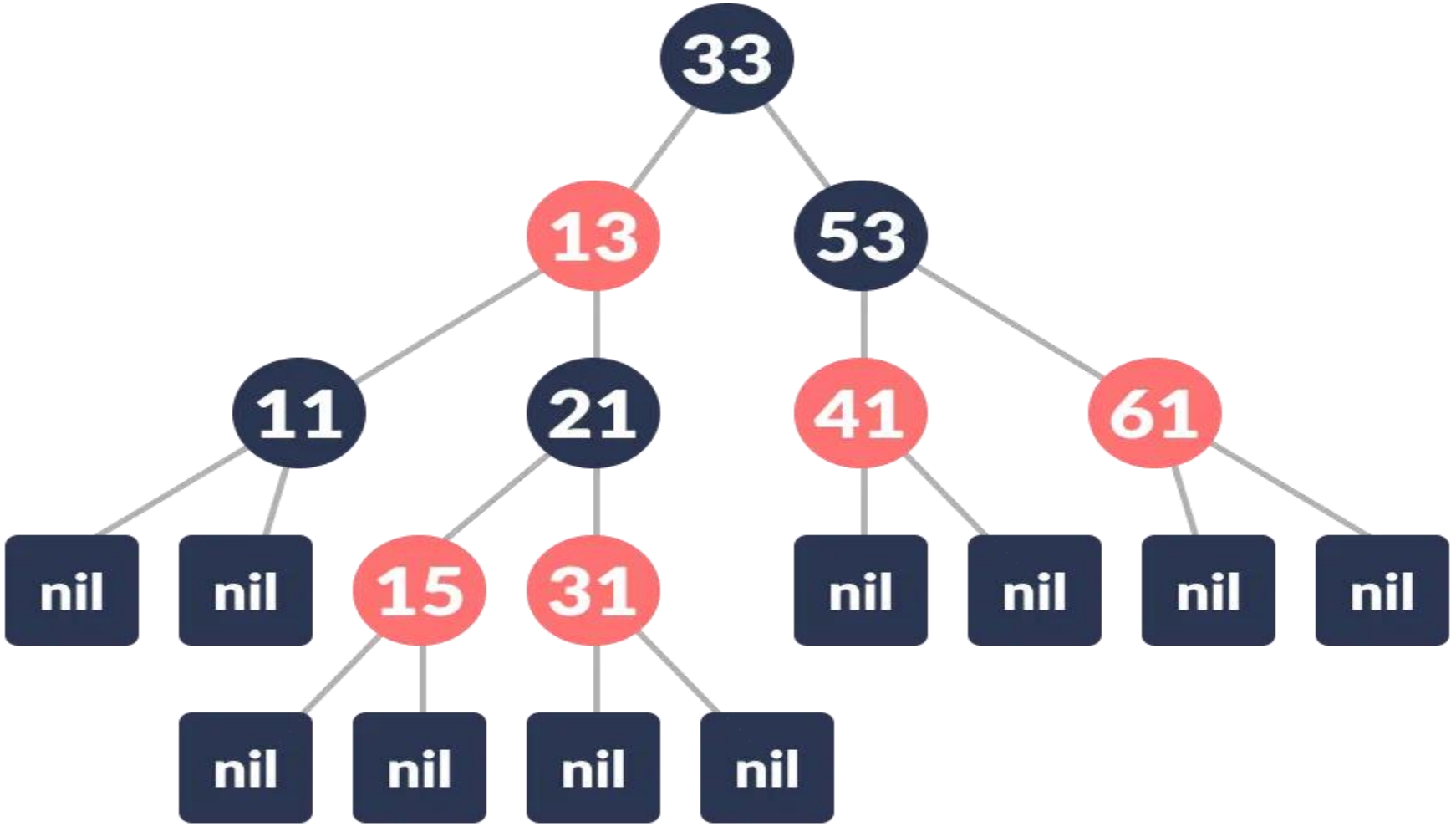
Red-Black Trees

Dr G.Kalyani
Department of Information Technology
VR Siddhartha Engineering College

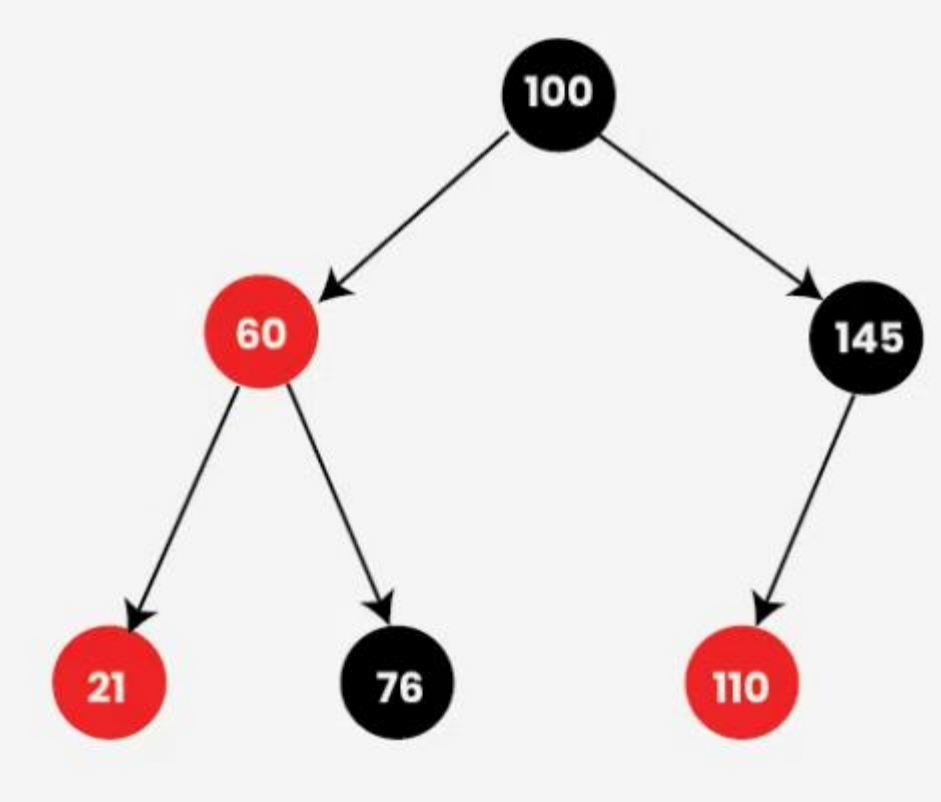
Introduction to Red-Black Trees

- **Definition:** A self-balancing binary search tree with additional properties.
- **Properties:**
  - **Node-color Property:** Every node should be colored either red or black.
  - **Root Property:** The root is always black.
  - **Red Property:** Red nodes cannot have red children.
  - **Black Property:** Every path from a root node to every leaf node has the same number of black nodes.
  - **Leaf Property:** All NULL nodes are black.

Example for Red-Black Tree



Is it Red-Black Tree?



Operations on Red-Black Trees

- The common operations
  - **Traversing**
  - **Searching**
  - **Insertion**
  - **Deletion**

Operations on Red-Black Trees

- **2 ways to perform the insertion/deletion operations in Red-Black Trees.**

- **Bottom-Up operations:**
  - **First perform the operation by searching in top down manner**
  - **Restore the properties by recursively travel back the tree from bottom.**

- **Top-Down Operations:**
  - **Rotations and recoloring are done while traversing down the tree to the insertion or deletion point.**

Bottom-Up Insertion Overview

Steps Involved:

- **Insert the node like in a BST.**
- **Color the new node red.**
- **Fix any violations of the red-black properties.**

Insertion: Fixing the violations Cases

Case 1: The new node is the root node.

- Recolor it to black.

Case 2: The parent node is black.

- The tree remains valid.

Case 3: Parent and parent sibling are both red.

- Recolor the parent and parent sibling to black
- Grandparent to red.

Case 4: Parent is red, parent sibling is black

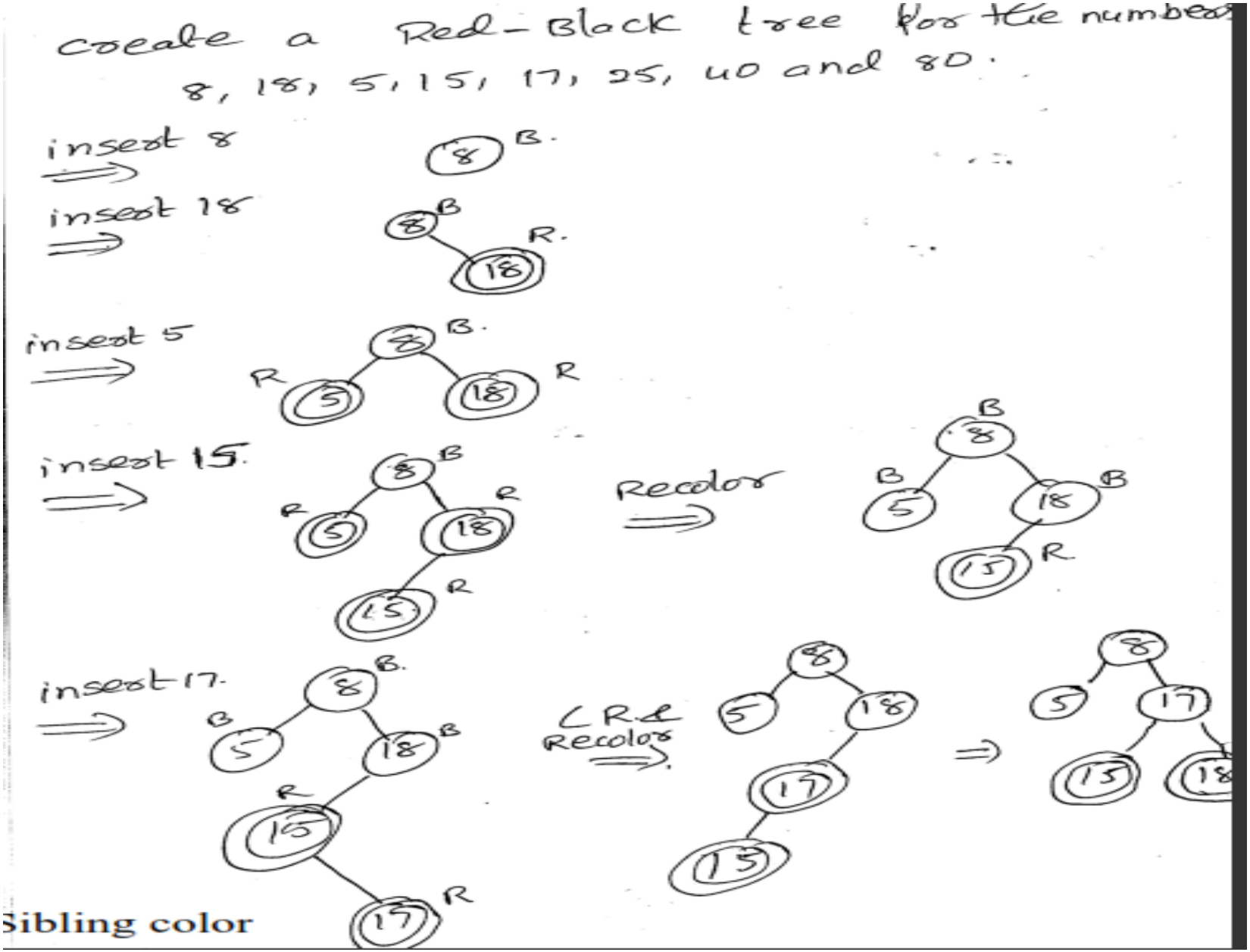
- Apply suitable Rotation
- After rotation assign parent to black and both the child's to red.

Example-1 of Insertion

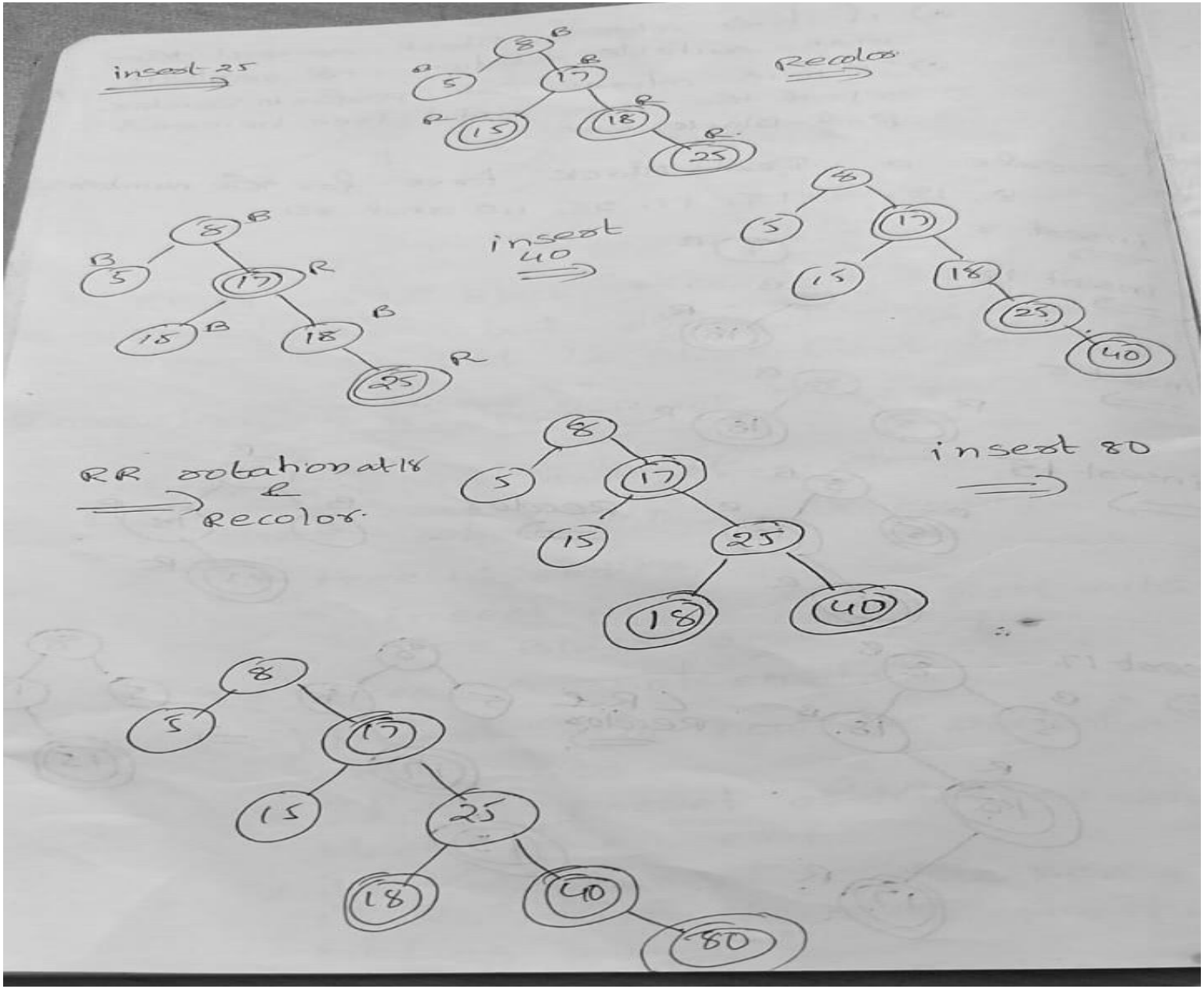
- **Create a Red-Black Tree for the following Data:**

  8, 18, 5, 15, 17, 25, 40, and 80

Example-1 of Insertion



Example-1 of Insertion

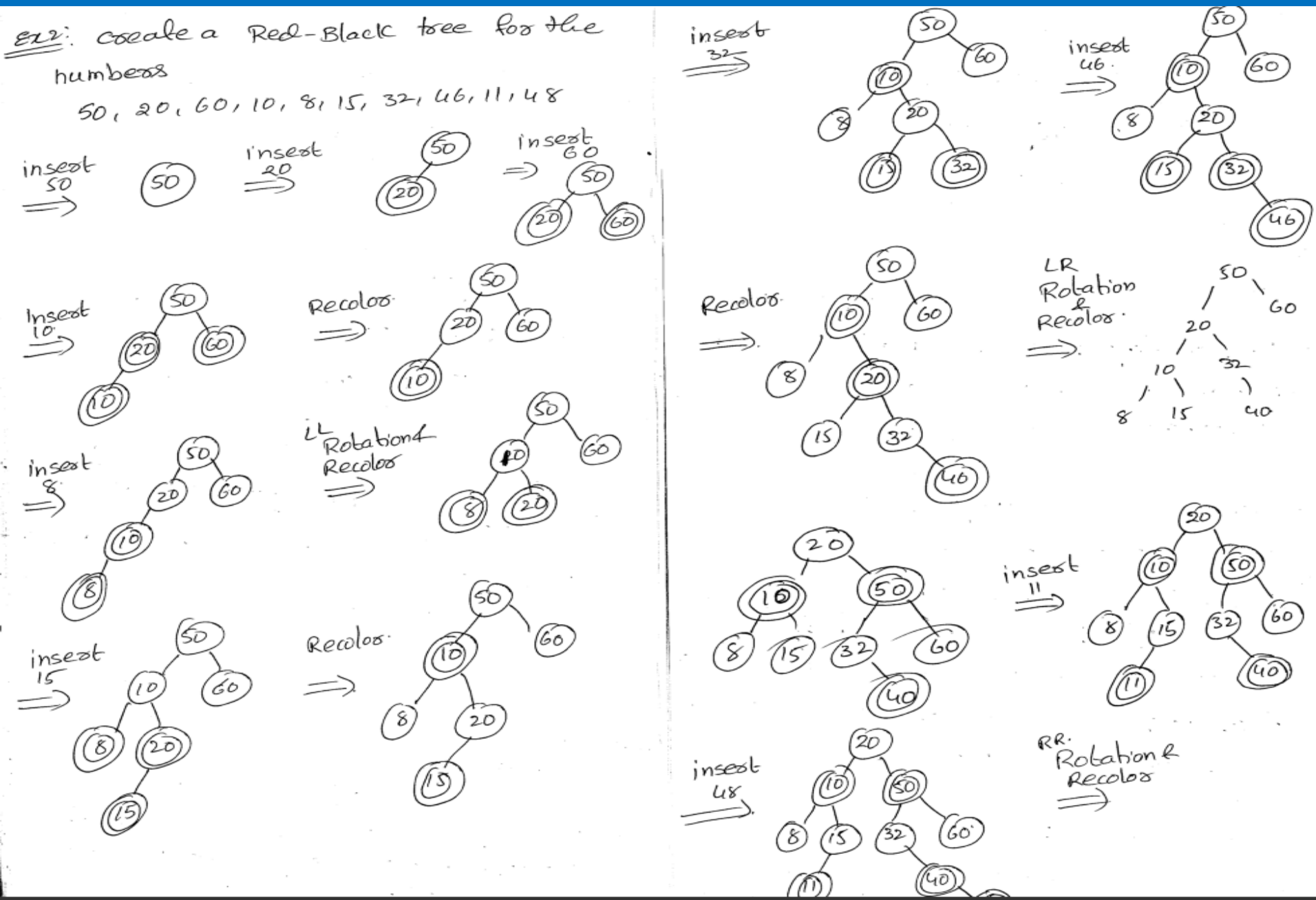


Example-2 of Insertion

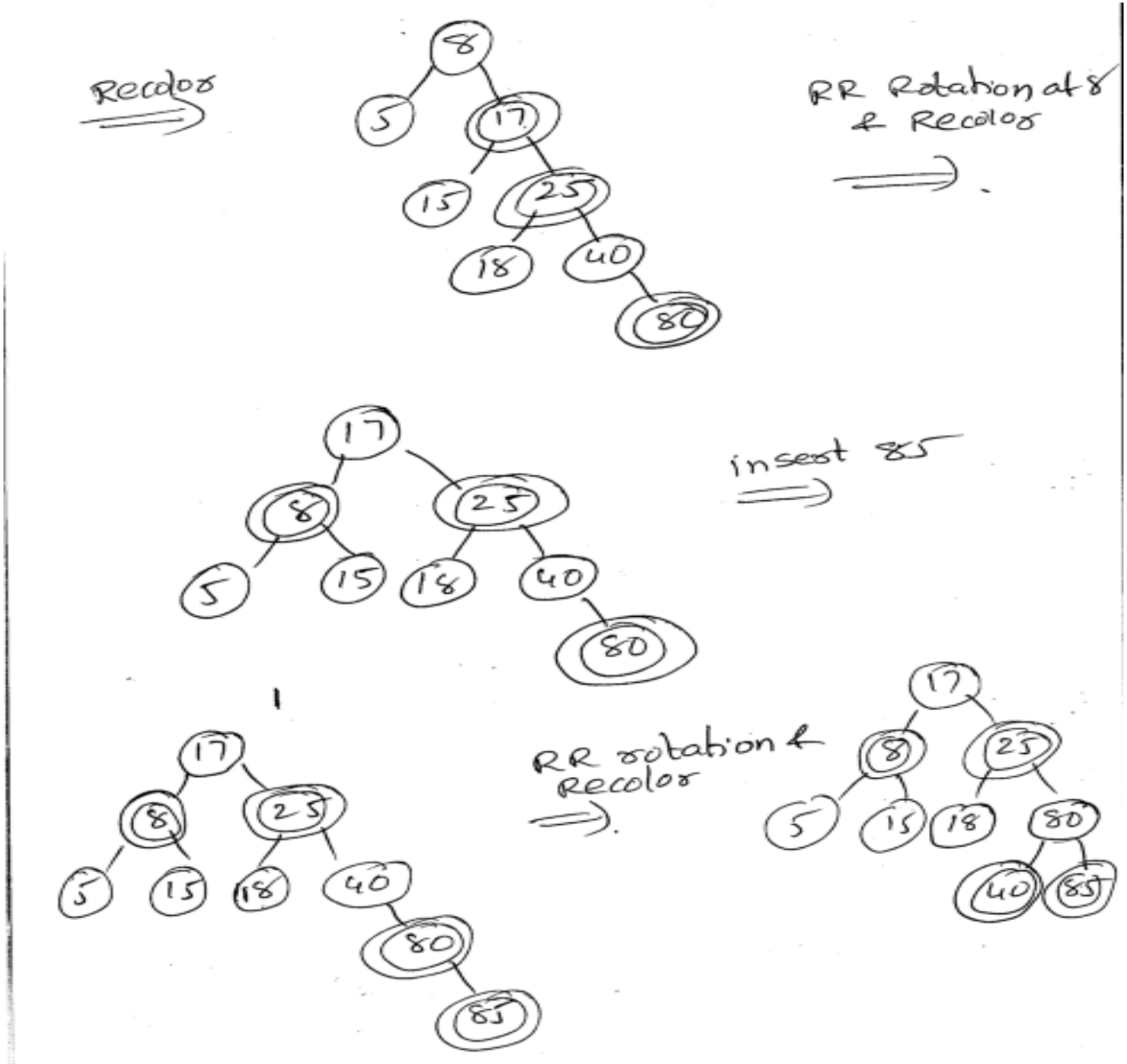
Ex2: Create a Red-Black tree for the numbers

50, 20, 60, 10, 8, 15, 32, 46, 11, 48

Example-2 of Insertion



Example-2 of Insertion



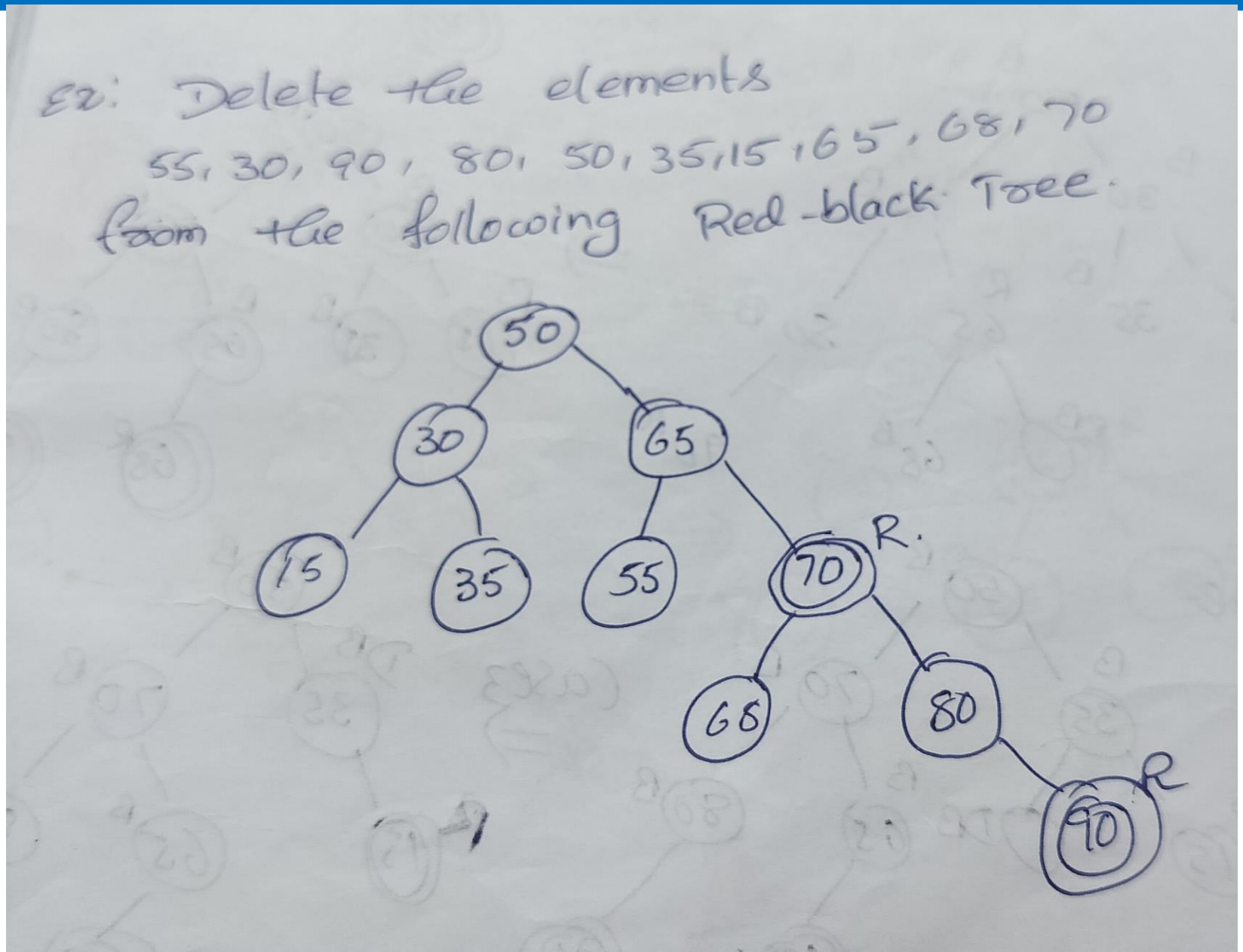
Bottom-Up Deletion Overview

- **Steps Involved:**
  - **Perform a standard BST deletion.**
  - **If the deleted node was red, the tree remains valid.**
  - **If the deleted node was black, fix any violations by recoloring and rotations.**

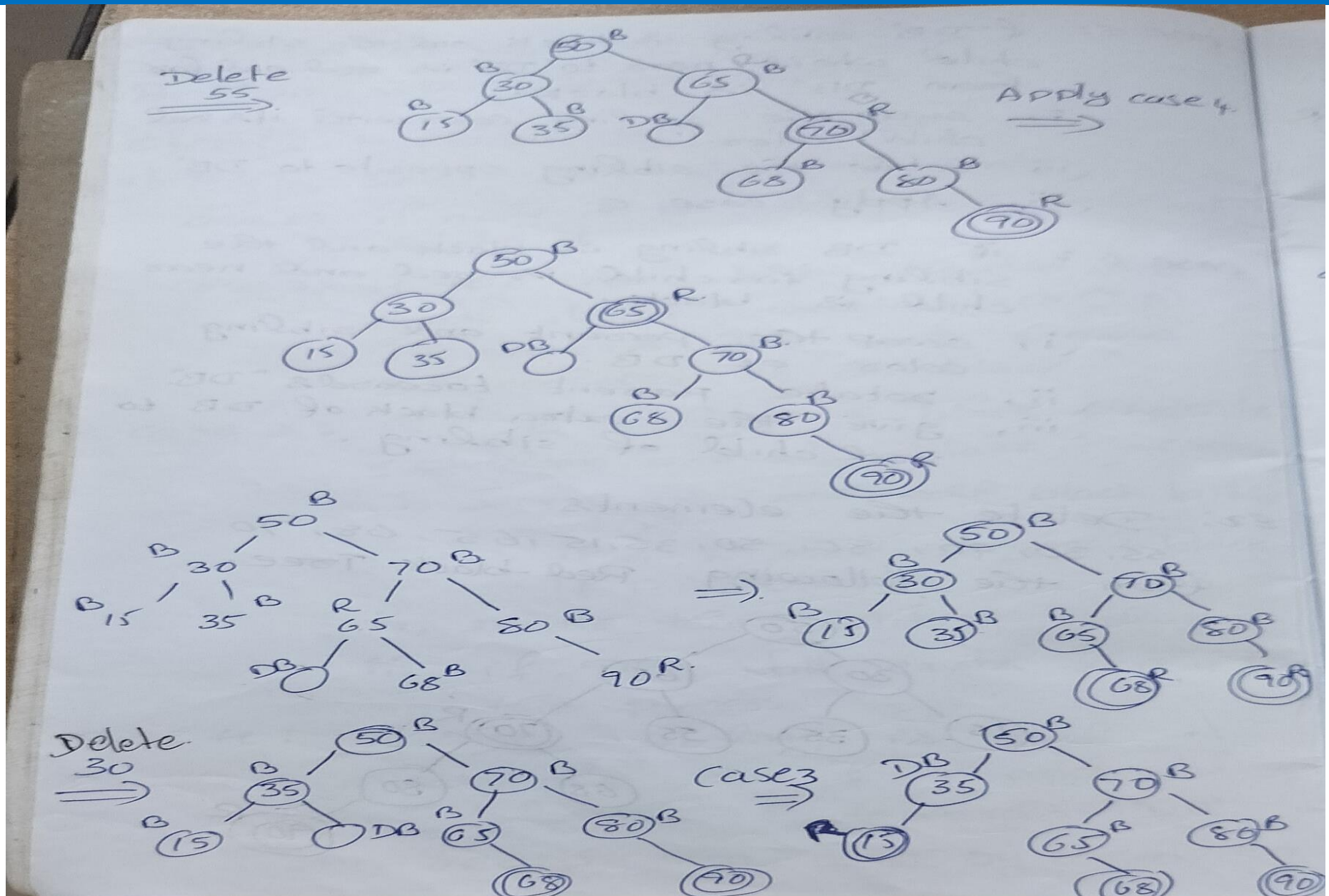
Deletion Cases in Detail

- if the node to be deleted is black **mark it as double black (DB).**
- **Case-1: If root is "DB"** → just remove DB **consider it as black.**
  Case-2: if "DB" sibling is black & both its children are also black.
  i. add extra black to its Parent (P)
  if P is red → becomes black. **if P is black → "DB".**
  ii. make the sibling Red.
  iii. if "DB" still exists repeat the cases.
  Case 3: if "DB" sibling is Red
  i. swap the color of P and sibling .
  ii. Rotate the P towards "DB".
  iii. reapply the cases by considering "DB".
  Case 4: if "DB" sibling is black and the sibling child who is near to " DB' is red and far from " DB " is black.
  i. swap the sibling color with its red child color.
  ii. rotate the sibling opposite to "DB".
  iii. Apply case 5.
  Case 5: if "DB" sibling is black and the sibling far child is red and near child is black.
  i. swap the parent and sibling color of DB.
  ii. rotate parent towards "DB".
  iii. give the extra black of DB to red child of sibling.

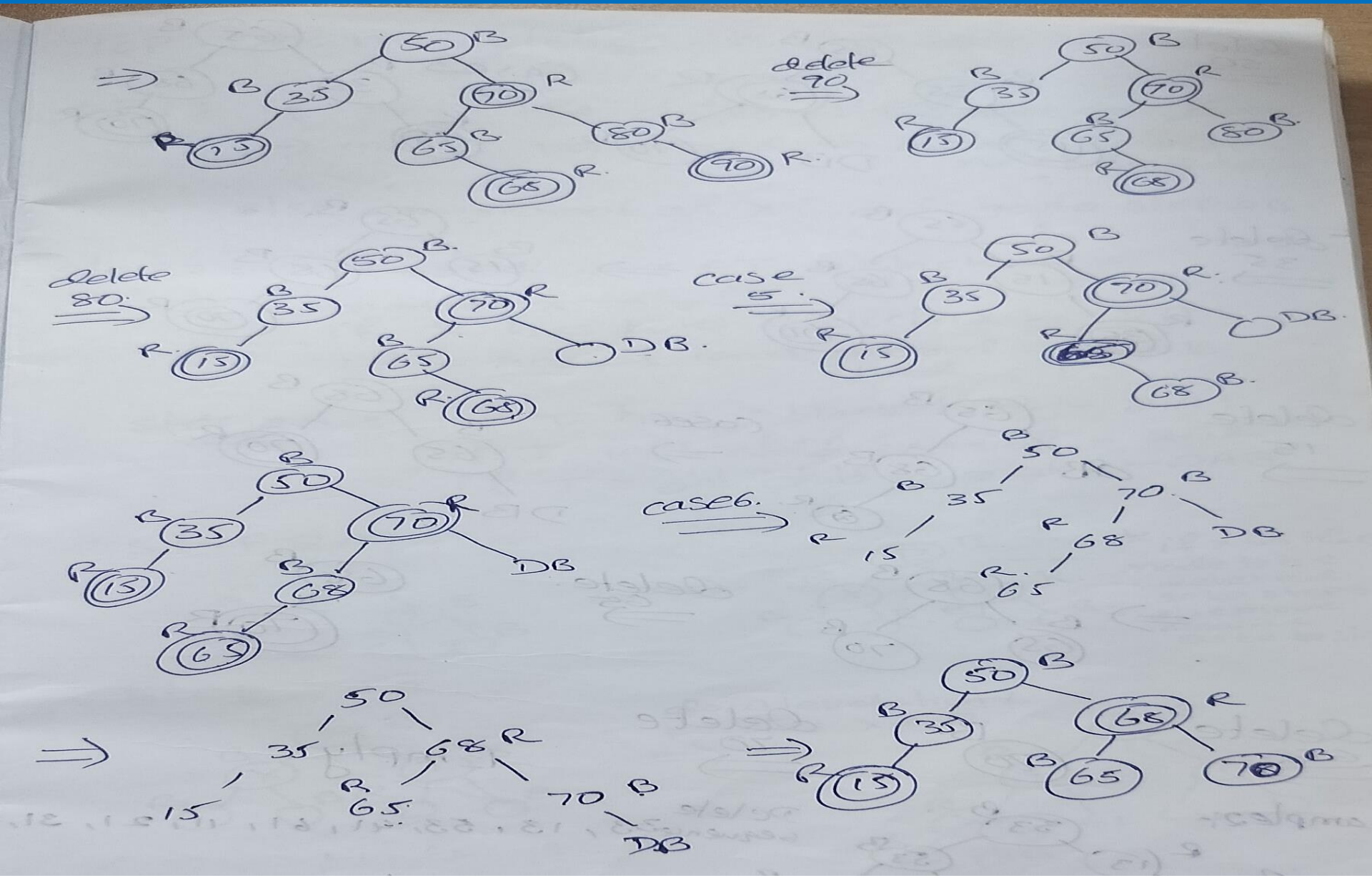
Example of Deletion



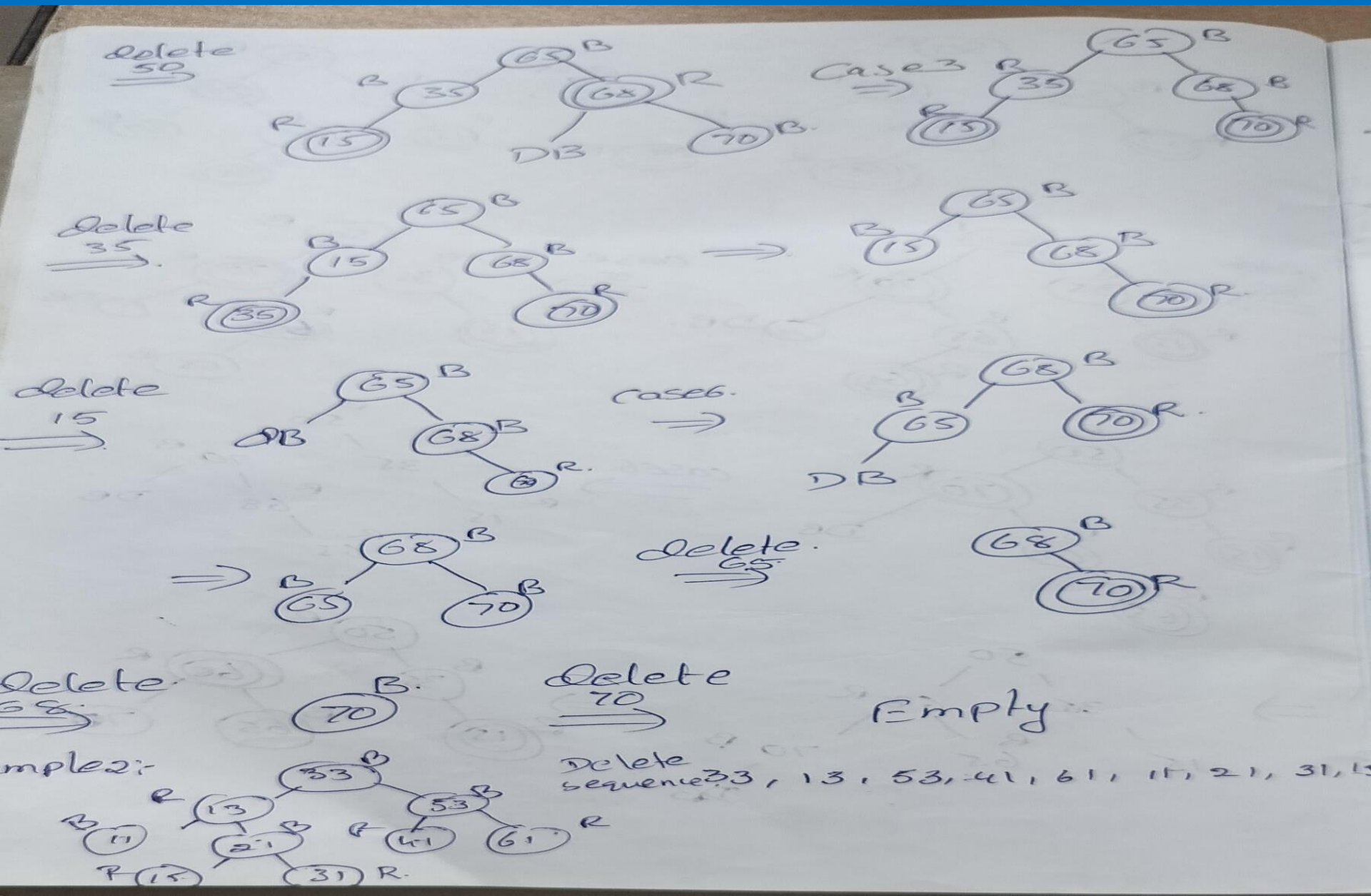
Example of Deletion



Example of Deletion



Example of Deletion



Advantages and Disadvantages

- **Advantages:**
  - Efficient insertions, deletions, and lookups.
  - More straightforward balancing rules compared to AVL trees.
- **Disadvantages:**
  - More cases than AVL trees in some operations.
  - Complexity in understanding the balancing rules.

Applications

- **Use Cases in Real-World Systems:**
  - **Databases**
  - **File systems**
  - **Memory management**
  - **Associative containers in programming languages (e.g., C++, Java)**

Summary

- Red-Black Trees ensure O(log n) time complexity for operations.
- Insertion and deletion involve fixing the tree using rotations and recoloring.
- The tree maintains balance through its properties, making it efficient for large datasets.

All birds find shelter during a rain. But eagle avoids rain by flying above the clouds. Problems are common, but **attitude makes the difference.**

– *Abdul Kalam*