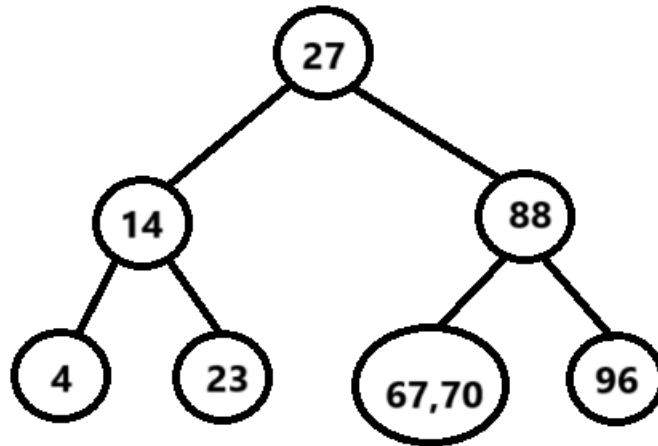


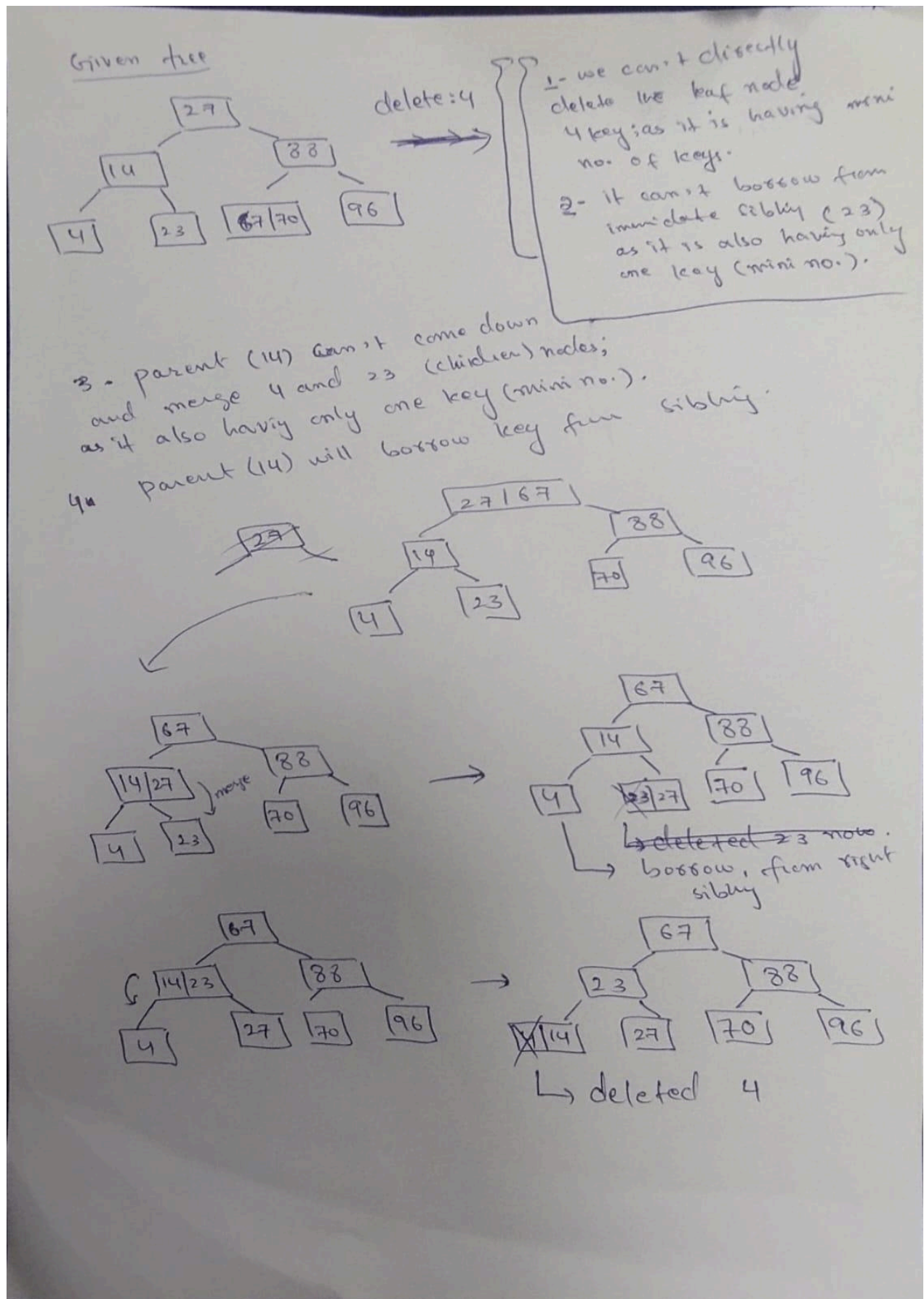
Name and ID: **Solution**Section: **BAI-3A**

Marks: 15

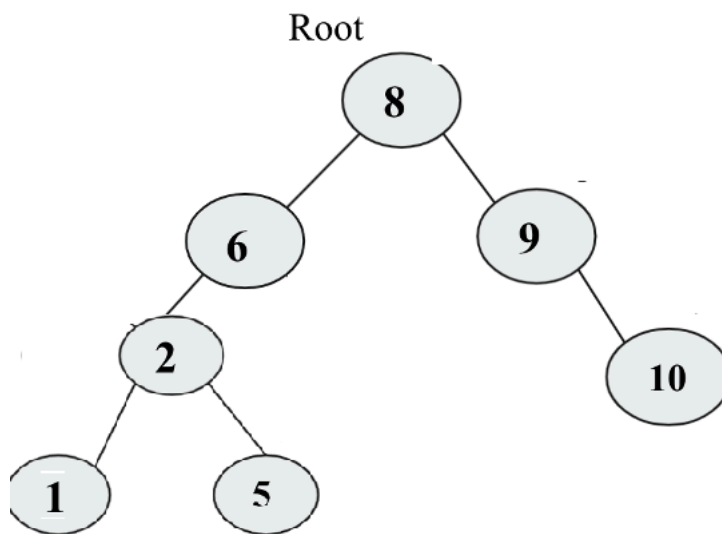
**Question#01 [3+5 Marks]**

- a. Given the 2-3 tree below, delete 4. Show each step of the process clearly and the final resulting tree by drawing each step clearly.

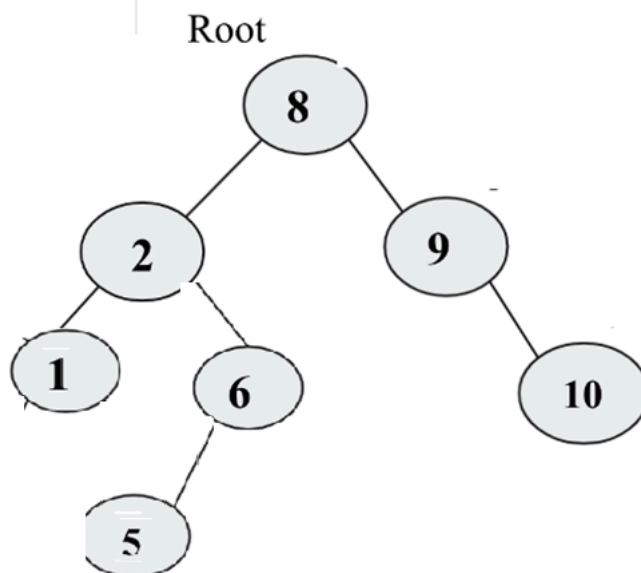




- a. Find which node is imbalance in the following tree and which AVL rotation is used to balance the node. Show rotation dry run and write C++ function for that rotation case.



Rotation case: LL imbalance (right rotation)



#### Right rotation

```
Node* LL_rotation(Node* node) {  
    Node* child = node->left_node;  
    node->left_node = child->right_node;  
    child->right_node = node;  
  
    node->height = max(get_height(node->left_node), get_height(node->right_node)) + 1;  
    child->height = max(get_height(child->left_node), get_height(child->right_node)) + 1;  
  
    return child;  
}
```

**Question#02 [7 Marks]**

```

#include <bits/stdc++.h>
using namespace std;

// To heapify a subtree rooted with node i
// which is an index in arr[].
void heapify(vector<int>& arr, int n, int i) {
    // Initialize smallest as root (for descending order)
    int smallest = i;

    // left index = 2*i + 1
    int l = 2 * i + 1;

    // right index = 2*i + 2
    int r = 2 * i + 2;

    // If left child is smaller than root
    if (l < n && arr[l] < arr[smallest])
        smallest = l;

    // If right child is smaller than smallest so far
    if (r < n && arr[r] < arr[smallest])
        smallest = r;

    // If smallest is not root
    if (smallest != i) {
        swap(arr[i], arr[smallest]);

        // Recursively heapify the affected sub-tree
        heapify(arr, n, smallest);
    }
}

// Main function to do heap sort
void heapSort(vector<int>& arr) {
    int n = arr.size();

    // Build heap (rearrange vector as a min-heap)
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

    // One by one extract an element from heap
    for (int i = n - 1; i > 0; i--) {
        // Move current root to end
        swap(arr[0], arr[i]);

        // Call min heapify on the reduced heap
        heapify(arr, i, 0);
    }

    // Reverse the array for descending order
    reverse(arr.begin(), arr.end());
}

// A utility function to print vector of size n
void printArray(vector<int>& arr) {
    for (int i = 0; i < arr.size(); ++i)
        cout << arr[i] << " ";
    cout << "\n";
}

// Driver's code
int main() {
    vector<int> arr = { 9, 4, 3, 8, 10, 2, 5 };

    // Function call
    heapSort(arr);

    cout << "Sorted array in descending order is:\n";
    printArray(arr);

    return 0;
}

```