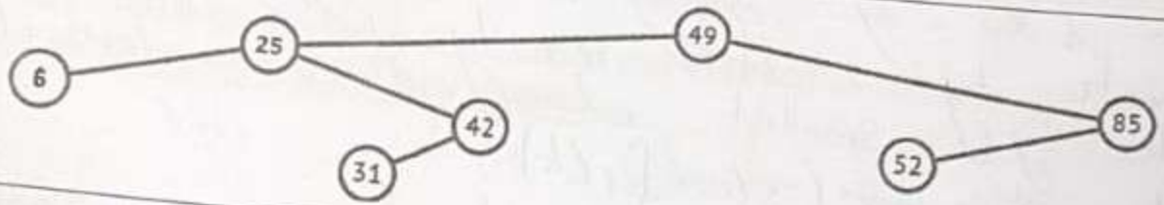
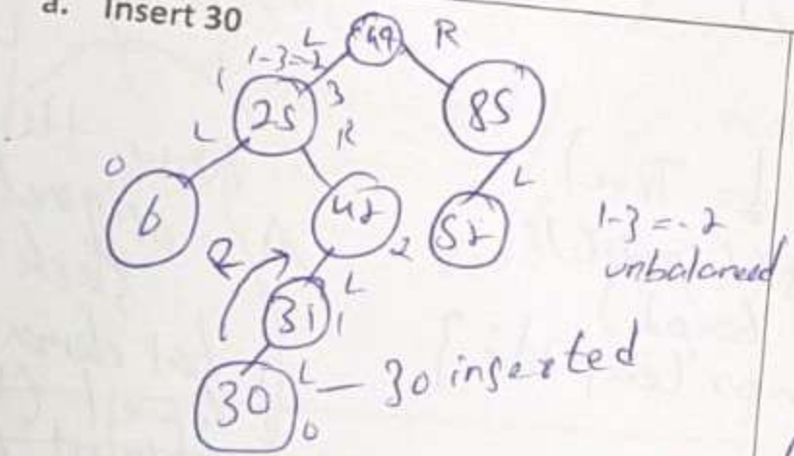
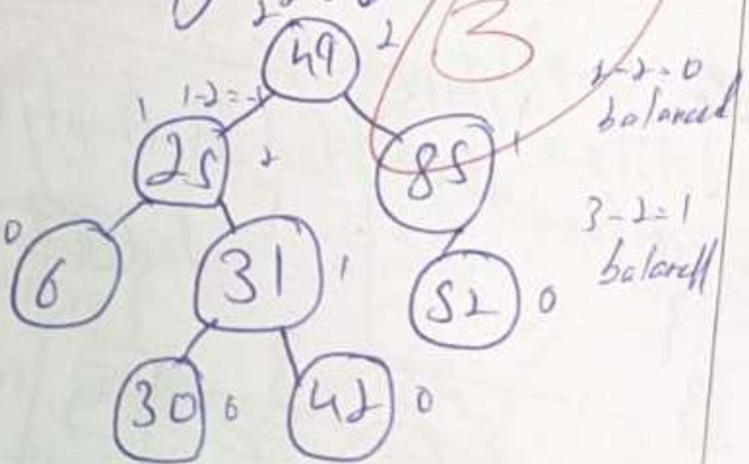
a) Considering the following tree as an AVL tree. Show the resulting AVL tree for each of the following cases. Show each step including unbalanced node, and any rotations or transformations applied. If double rotation is applied then also show the intermediate steps. **(Marks: 5+3+2)**



a. Insert 30

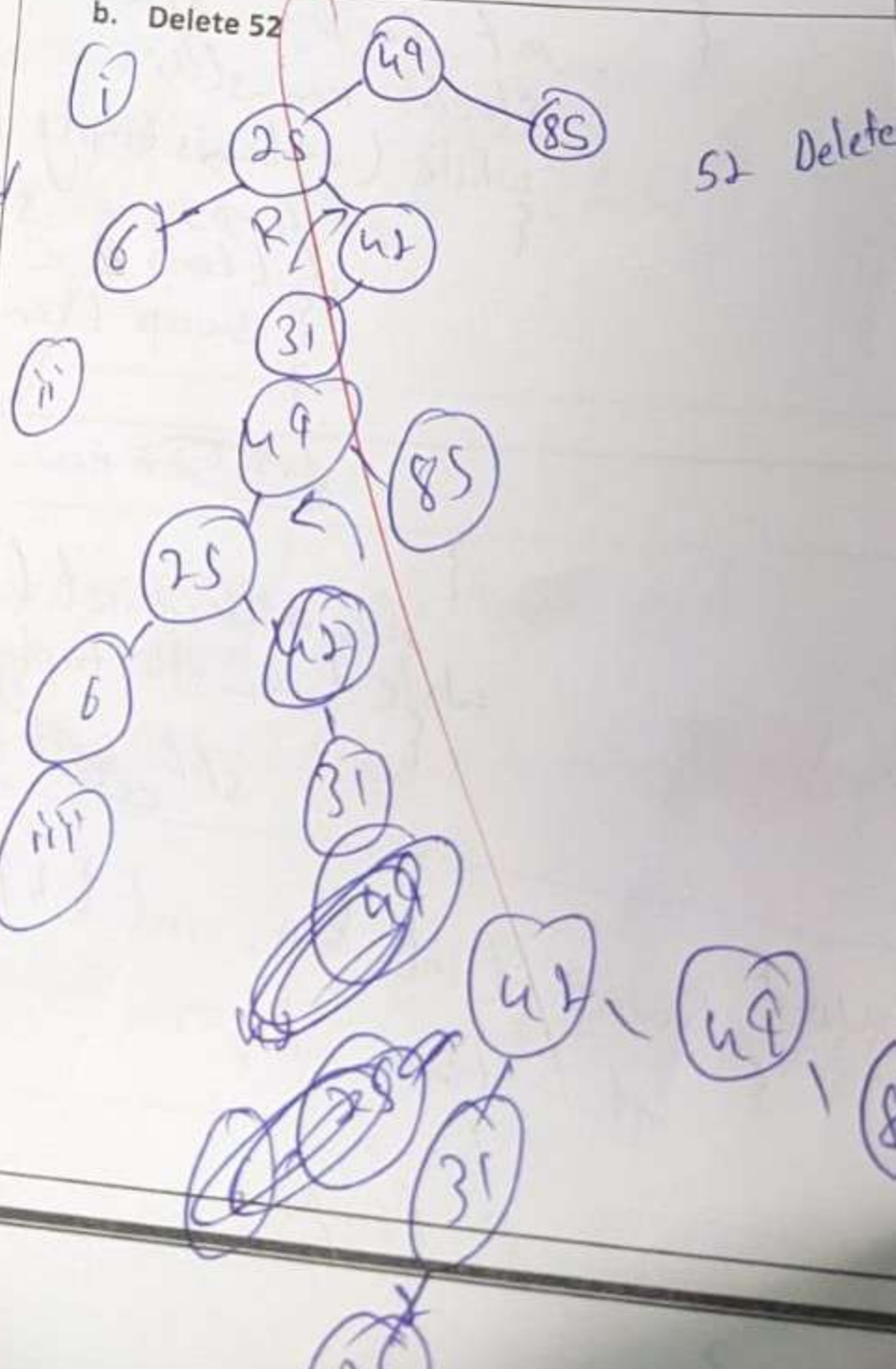


→ Using Right Rotation



Steps = 1 [Right rotation on 31 & 42 nodes]

b. Delete 52



b) For each of the scenarios given below, suggest the most appropriate data structure chosen from list and **give appropriate reason of your choice**.
(Arrays, linked-list (single, double, circular), Queue, Stack, tree(BST, AVL))

To implement a dictionary of words in a language. Tree (AVL)	languages have a large amount of words and searching takes time, AVL reduces it to max O(log n)
Implementing a music playlist with a shuffle and repeat feature. Double circular Linked list	By using it, you can move back and forth in O(1) time and also shuffle easily.
Tracking a history of web pages visited by a user in a web browser. stack	Because the LIFO order allows to see the most recent visited page by popping in O(1)

c) Suppose we want to write a function that requires LIFO order (stack) and memorize the minimum data element of the stack in O(1). If the minimum value is popped from the stack then new minimum should be compued in O(1) time. This can be done using a temporary stack. Explain how a tempoary stack can be utilized to accomplish this task.

…ion 2: [CLO: 3]

(Marks: 10)

… a <u>recursive</u> C++ function ConstructBST() for an integer based binary search tree that takes an array *preorder* … its size *n* as parameter. The array preorder stores the data of a BST traversed in pre-order. Your task is to …nstruct and return the root of that BST. You may need to pass some other parameters to complete this task. …ompute the time complexity of your function. Note that less credit will be given to less efficient solutions.

Sample Input	Sample Output
7,6,3,1,4,9	7 (left child 6, right child 9); 6 (left child 3); 3 (left child 1, right child 4)
1,3,5,7	1 \ 3 \ 5 \ 7 (each a right child)

```
        7
       / \
      6   9
     /
    3
   / \
  1   4
```

```
1
 \
  3
   \
    5
     \
      7
```

4

What extra parameters you want to pass to this function and what is their significance? Explicitly write the names and types of those parameters and their purpose.

You need the function to know which element is being stored right now. Passing by refrence helps move & root, change value each child. updates

Give complete function Definition

Function Prototype

```
Node* ConstructBST ( int * arr , int n) Node * root
{        int current_index = 0;
   return ConstructBST( arr, n, Current_index
}
```

Base case

```
if ( curr ≥ n )
{
    return root;
}
if ( root == NULL)
{
```

Recursive case

```
    Node * temp = new Node();
    temp → data = arr[curr];
    curr++;
    root = temp;
    return root;
}
else if ( root → data > arr[curr]
{
    Construct BST (arr, n
    curr++;
}
else if ( root → data ≤ arr[curr]
{
    Construct BST (arr, n, cu
    curr++;
}
curr++; //As it is passed
return root;
else if ( root → data == arr[
{
    // Duplicate
    curr++; return
}
return root;
}
```

Time Complexity

Time Complexity of this code is O(N) if it