**LM Data Structures, Algorithms, and Databases (34140, 34141, 34139, 36989)**

**Exercise Sheet Solution**

**Week 02**

Q1. Assume a perfectly balanced binary search tree,

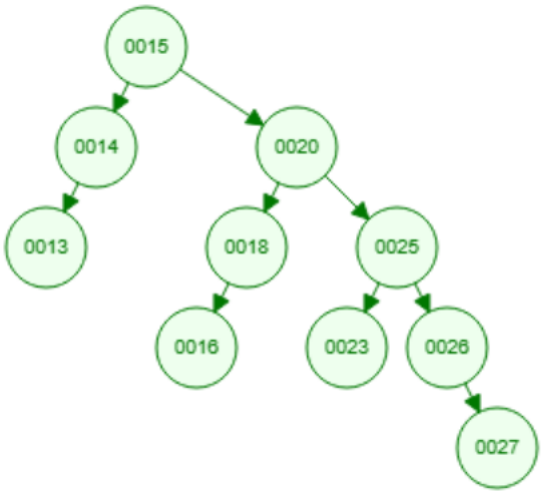
a. If the height of a tree is given as 10, what is the total number of nodes that could be present in this tree? Answer: $2^h-1 = 2^{10}-1 = 1023$

a. What is the height of a tree with a total of 1 trillion nodes?
Answer: $\log_2(n+1) = \log_2(1 \text{ trillion}+1) = \log_2(10^{12}+1) = 39.8 = 40$

Q2. Construct a Binary Search Tree (BST) by sequentially **inserting** the values

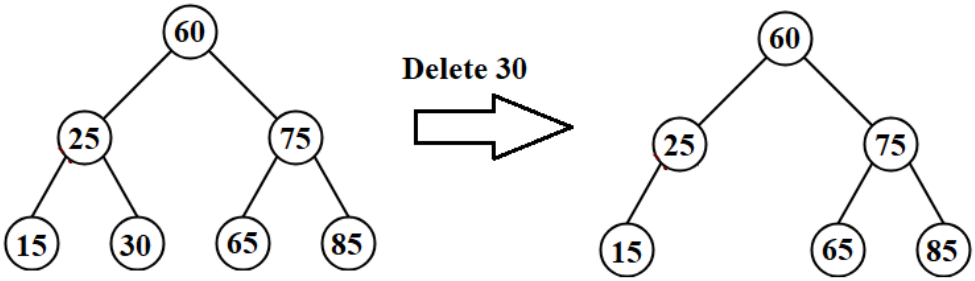
**Part A:** 15, 20, 25, 18, and 16.


**Part B:** 15, 20, 25, 18, 16, 14, 26, 27, 13, 23.

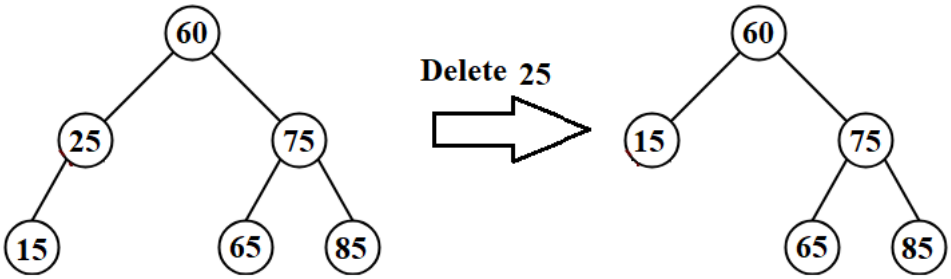


Q3. Show the effect of **delete** operation on the following Binary Search Tree (BST).

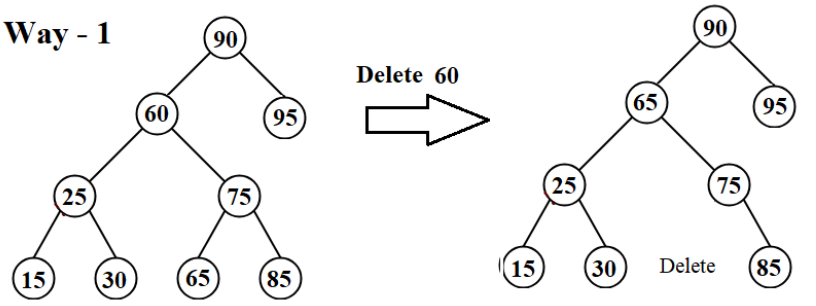
**Part A: Delete 30**

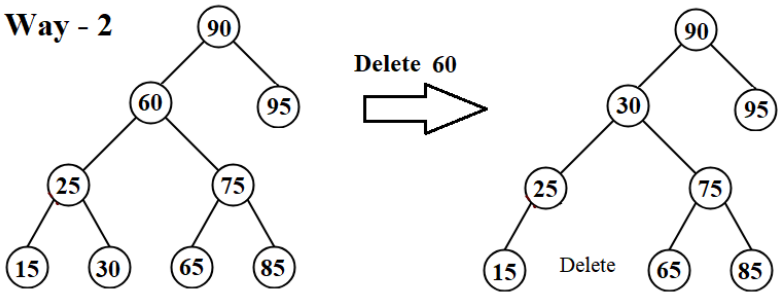



**Part B: Delete 25**




**Part C: Delete 60**



Q4. Describe the results of three types of tree traversal (in-order, pre-order, and post-order) on the following binary tree:


Provide the sequence of nodes visited for each traversal method: in-order, pre-order, and post-order.

Answer:

In-Order Traversal: 4 2 5 1 6 3 7
Pre-Order Traversal: 1 2 4 5 3 6 7
Post-Order Traversal: 4 5 2 6 7 3 1

Q5. Provide pseudocode to calculate the height of a binary tree.

```
function treeHeight(node):
    # Base case: If the node is null, the height is 0
    if node is null:
        return 0

    # Recursive case: Calculate the height of the left and right subtrees
    leftHeight = treeHeight(node.left)
    rightHeight = treeHeight(node.right)

    # The height of the tree is the maximum height of its left and
    # right subtrees, plus 1 for the current level
    return max(leftHeight, rightHeight) + 1
```

Q6. Provide pseudocode to count the total number of nodes in a binary tree.

```
function countNodes(node):
    if node is null:
        return 0

    # Recursive case: Count the current node and nodes in left and right subtrees
    return 1 + countNodes(node.left) + countNodes(node.right)
```

Q7. Provide pseudocode for counting the number of leaf nodes in a binary tree.

```
function countLeafNodes(node):
    if node is null:
        return 0

    if node has no left and no right children:
        return 1

    leftLeafCount = countLeafNodes(node.left)
    rightLeafCount = countLeafNodes(node.right)

    return leftLeafCount + rightLeafCount
```

Q8. Given below are the equations for the Insertion algorithm of binary search trees. Fill in the blank.

```
insert(x, NIL) = Fork(x, NIL, NIL)

insert(x, Fork(x, l, r)) = Fork(x, l, r)

insert(x, Fork(y, l, r)) = Fork(y, insert(x, l), r), if x < y

insert(x, Fork(y, l, r)) = _____________________, if x > y
```

Answer: `Fork(y, l, insert(x, r))`

Q9. For deleting the value at the root of a tree that has two nonempty subtrees, the following equation was used in the lecture:

```
delete(x, Fork(x, l, r)) = Fork(smallest(r), l, removeSmallest(r))
```

This replaces the value at the root by the smallest value of the right subtree.

One can also replace the value at the root by the largest value of the left. Write the formula for this alternative.

Answer: `delete(x, Fork(x, l, r)) = Fork(largest(l), removeLargest(l), r)`

Q10. Consider modifying the algorithms for binary search trees so that multiple copies of values can be inserted. How should the equation for insert be modified?

```
insert(x, Fork(x, l, r) = _______________
```

Answer: `Fork(x, insert(x, l), r)` Or `Fork(x, l, insert(x, r))`