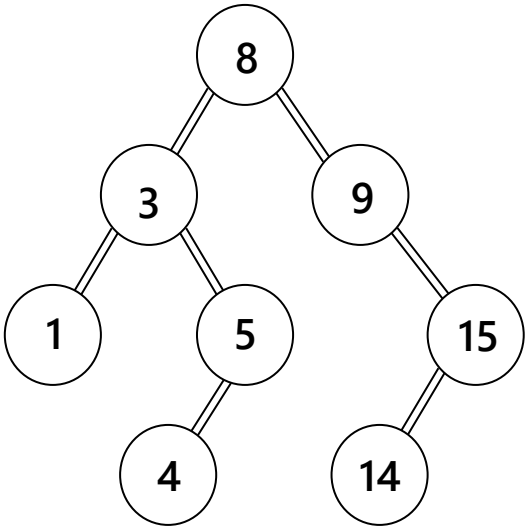
Trees

For the following problems, assume that the tree considered is the one shown below unless stated otherwise.



1. **[TA]** Show how the tree is represented in an array.
2. **[TA]** Using the array representation and for specific node, find explicit formulas for finding the index of the parent node and the indices of the two children nodes.
3. **[TA]** Show the output when pre, post and in-order traversals are applied on the tree.
4. **[TA]** Show in pseudocode, the recursive programs of the 3 traversals, mentioned in the previous question, for a certain binary tree.
5. **[TA]** Define the difference between a general tree, a binary tree and a binary search tree. State why the tree considered is a binary search one.
6. **[TA]** Come up with toy numbers that you may insert in the tree so that it becomes a complete binary tree.
7. Show that the height of a complete binary tree is $\boldsymbol{O}(\log N)$ where N is number of nodes in this tree.
8. Given a sequence of numbers: `11, 6, 8, 19, 4, 10, 5, 17, 43, 49, 31`, draw a binary search tree by inserting the above numbers from left to right.
9. **[TA]** Try to delete the nodes with following values: `5, 9, 8` and show your tree afterwards. Refer to original tree after deleting each node separately.

10. Show in pseudocode, how to search for a certain value in binary search tree. Let your work be under a procedure called `search(Tree, value)` where Tree is the root node and the `value` is your target you are searching for.

11. In pseudocode, write down an implementation of procedure `delete(Tree, value)` that deletes node with `value`, if exists, given root node `Tree`. Consider the `search()` procedure in previous question directly without re-implementation.

12. [**TA**] Write pseudocode for `getHeight(Tree)` procedure that prints the height of binary tree given its root node `Tree`.

13. Write pseudocode for `leavesNum(Tree)` procedure that prints the number of leaf nodes of binary tree given its root node `Tree`.

14. Given an array of *N* numbers, show how to sort them using a binary search tree. Write your algorithm in pseudocode.

15. A "root-to-leaf path" is a sequence of nodes in a tree starting with the root node and proceeding downward to a leaf node. Given a binary tree root node `Tree`, consider the "root-to-leaf" path having the maximum sum of nodes. Return this sum. The method signature is `maxPathSum(Tree)`. Implement the method in **Python**.

16. Change a binary tree so that the roles of the left and right pointers are swapped at every node. So, for example, the tree below

Given a binary tree root node `Tree`, implement the method with the signature `mirror(Tree)` in **Python** that achieves the objective described.

Binary Heaps (Part I)

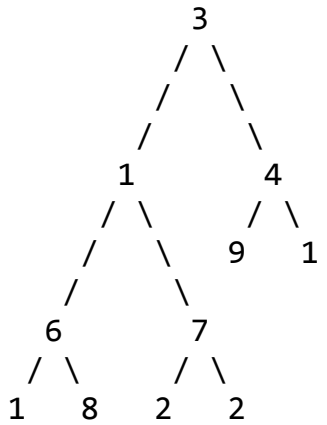
1. Describe what is meant by a maximum binary heap. Compare between minimum binary heap, maximum binary heap and binary search tree. Can a binary tree be both a minimum binary heap and a binary search tree at the same time for number of nodes > 2?

2. Below is the pseudocode for `heapify()` method that converts an array `arr` representation of complete binary tree into maximum heap. Show that time complexity for such method is $\mathcal{O}(n)$.

```
heapify(arr, n):
    for i = n//2 - 1 to 0:
        maxHeapify(array, i, n)

maxHeapify(arr,parentIndex, n):
    while True:
        left = parentIndex * 2 + 1
        right = parentIndex * 2 + 2
        largest = parentIndex
        if left < n and arr[left] > arr[largest]:
            largest = left
        if right < n and arr[right] > arr[largest]:
            largest = right
        if i == largest:
            break
        swap(arr[largest], arr[parentIndex])
        parentIndex = largest
```

3. Apply the `heapify()` method described in previous question to convert the complete binary tree below into maximum heap. Show how the tree changes along the different steps.



Good Luck