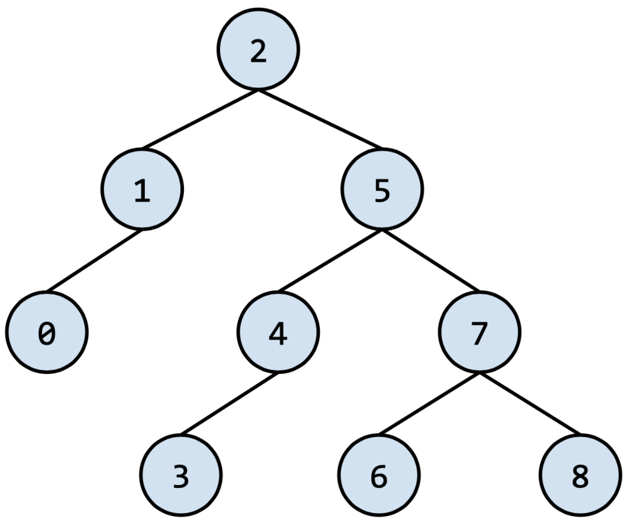
1 Traversals

Taken from Summer 2021 MT2.

a) (1 Point). Given the tree below, give its **preorder**, **inorder**, **postorder**, and **level order** traversals. Format each traversal as a space separated list, e.g. 1 2 3 4.



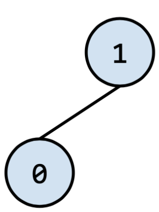
Preorder: ____

Inorder: ____

Postorder: ____

Level Order: ____

b) (1.5 Points). It's possible that 2 *different* traversals on the same tree produce the same ordering. For instance, on the tree below,



notice the postorder and inorder traversals are both 0 1.

Looking at the tree from the previous part, notice that none of the traversals produce the same ordering. However, we may be able to remove **some leaf nodes** from the tree such that running **two** of the above traversals on the modified tree produce the same ordering. Prioritizing removing the **minimum** amount of leaf nodes possible, select the leaf nodes that need to be removed and the two traversals that become the same. If you select more leaf nodes that the minimum, you won't receive any credit. If it isn't possible, select "impossible" for both options below.

Leaf nodes to remove: ◯ 0 ◯ 3 ◯ 6 ◯ 8 ◯ impossible

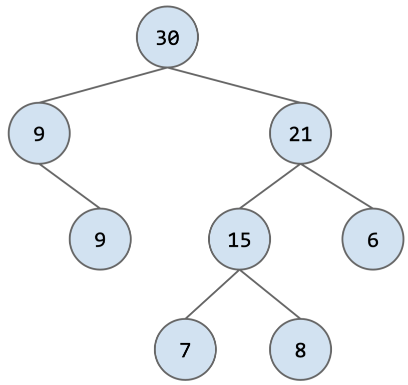
Traversals: ◯ Preorder ◯ Inorder ◯ Postorder ◯ Level Order ◯ impossible

2 Binary Trees

For each of the following question, use the following `Tree` class for reference.

```java
public class Tree {
    public Tree(Tree left, int value, Tree right) {
        _left = left;
        _value = value;
        _right = right;
    }
    public Tree(int value) {
        this(null, value, null);
    }
    public int value() {
        return _value;
    }
    public Tree leftChild() {
        return _left;
    }
    public Tree rightChild() {
        return _right;
    }
    private int _value;
    private Tree _left, _right;
}
```

(a) Given a binary tree, check if it is a sum tree or not. In a sum tree, the value at *each* non-leaf node is equal to the sum of its children. For example, the following binary tree is a sum tree:




```
public boolean isSumTree(Tree t) {
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
}
```

(b) Given a binary tree with distinct parts, an input list, and an empty output list, add all the elements in the input list to the output list that appear in the tree. The elements in the output list should be ordered in the same order that would be returned from an inorder traversal.

For example, for the tree in Q.2(a) assuming it has only distinct parts, if the input list is [15, 9, 8, 30, 6], then after the operation the output list should be [9, 30, 15, 8, 6].

```
public static void sortRelative(Tree t,
                               List<Integer> inputList,
                               List<Integer> outputList) {
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
}
```

3 An Unintuitive Game

Sumer challenges Sohum to the following game. In this game, there is a maximizing player and a minimizing player. Both players take turns adding numbers to the end of the sequence. The maximizing player wants to maximize the **last** number in the sequence, and the minimizing player wants to minimize it.

On a player's turn, they take the previous number in the sequence and create the next number by either:

- floor dividing it by 2
- multiplying it by 3 and adding 1

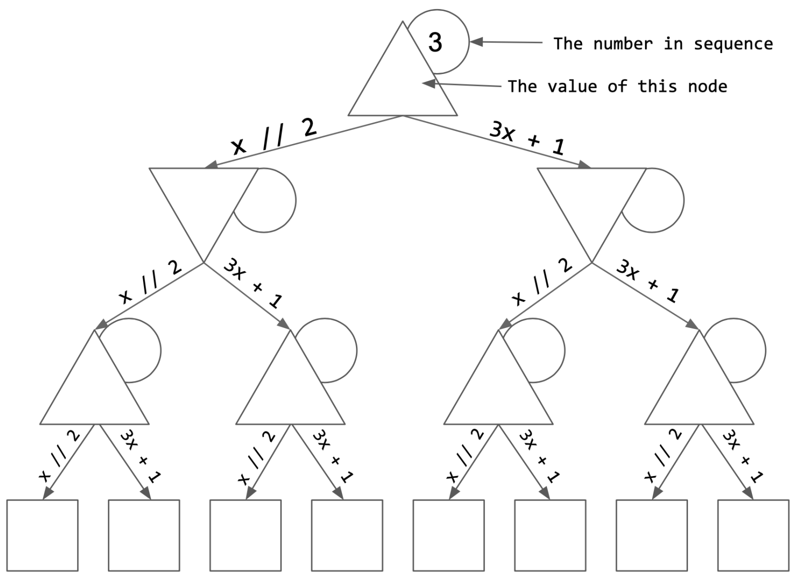
The sequence starts with 3, and can only contain numbers in the range 1 - 5. If a number goes out of bounds after an operation, it wraps around. For example, floor dividing 1 by 2 is zero, which wraps around to 5. Finally, the maximizing player will always start and get two turns. The minimizing player will get one turn.

Here is an example of a sequence.

1. We start with **3**.
2. The maximizer chooses to multiply 3 by 3 and add 1, giving us 10, which wraps around to **5**.
3. The minimizer chooses to floor divide 5 by 2 to get **2**.
4. The maximizer chooses to multiply 2 by 3 and add 1, giving us 7, which wraps around to **2**.
5. The last number in the sequence is ***2***.

(a) Fill in the minimax game tree for the following game. Typically in game trees only store the **value** of each node, but for this game it will be helpful to keep track of the current number in the sequence, so an additional circle has been given for that.

Note that no circle is written besides the leaf nodes because the value of each leaf node is the current number! Also note that we wrote 3 in the first circle to get you started (since the sequence starts with 3).



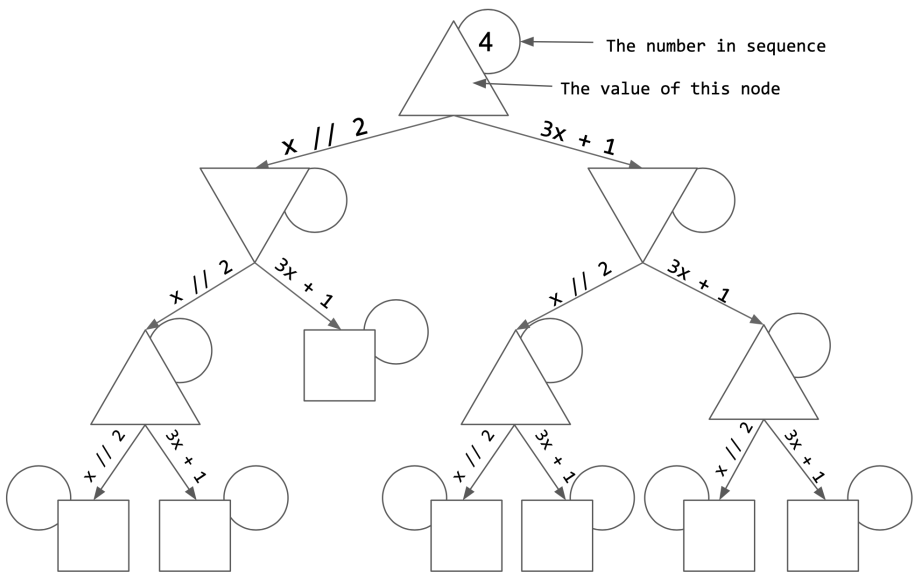
(b) Assuming both players play optimally, what is the last number in the sequence?

(c) Using the game tree from the part a, which branches can be pruned with alpha beta pruning? Cross out the branches, if any, in the previous tree.

(d) Now, suppose that we keep the game *exactly* the same but we change two things:

1. We start with 4.
2. The maximizer **wins** if there is a duplicate in the sequence and the minimizer wins if the numbers are unique.

Fill in the game tree for the modified game. Note that a circle is written beside each leaf node because the value of each leaf node isn't the current number! Hint: Use ∞ and $-\infty$ to represent the maximizer winning and the minimizer winning, respectively.



(e) Assuming both players play optimally in this modified game, who wins?

(f) Using the game tree from part c, which branches can be pruned with alpha beta pruning? Cross out the branches, if any, in the previous tree.