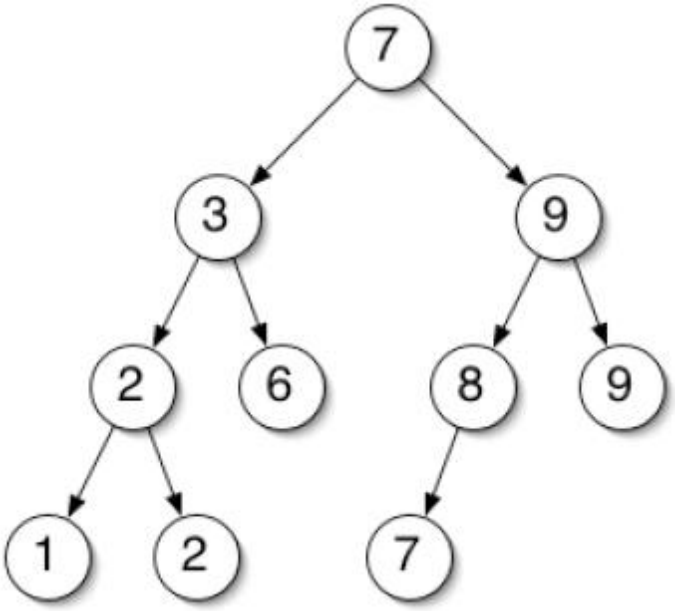
1. Tree traversals

Consider the following pair of recursive algorithms calling each other to traverse a binary tree.

```
Algorithm weirdPreOrder(treeNode n)
    if (n != null) then
        print n.getValue()
        weirdPostOrder( n.getLeftChild() )
        weirdPostOrder( n.getRightChild() )
```

```
Algorithm weirdPostOrder(treeNode n)
    if (n != null) then
        weirdPreOrder( n.getRightChild() )
        weirdPreOrder( n.getLeftChild() )
        print n.getValue()
```

a) Write the output being printed when weirdPreOrder(root) is executed on the following binary tree:



b) Write the output being printed when weirdPostOrder(root) is executed on the same binary tree (above).

c) Consider the binary tree traversal algorithm below:

```
Algorithm queueTraversal(treeNode n)
Input: a treeNode n
Output: Prints the value of each node in the binary tree rooted at n
Queue q ← new Queue();
q.enqueue(n);
while (! q.empty() ) do
    x ← q.dequeue();
    print x.getValue();
    if ( x.getLeftChild() != null ) then q.enqueue( x.getLeftChild() );
    if ( x.getRightChild() != null ) then q.enqueue( x.getRightChild() );
```

Question: write the output being printed when queueTraversal(root) is executed on the tree above. This is the equivalent of what traversal method seen previously in class?

d) Consider the binary tree traversal algorithm below:

```
Algorithm stackTraversal(treeNode n)
Input: a treeNode n
Output: Prints the value of each node in the binary tree rooted at n
Stack s ← new Stack();
s.push(n);
while (! s.empty() ) do
    x ← s.pop();
    print x.getValue();
    if (x.getRightChild() != null) then s.push(x.getRightChild());
    if (x.getLeftChild() != null) then s.push(x.getLeftChild());
```

Question: Write the output being printed when stackTraversal(root) is executed on the tree above. This is the equivalent of what traversal method seen previously in class?