Muangkong Yang
CSCD320 – HW3

In-order = {9, 5, 1, 7. 2, 12, 8, 4, 3, 11}
Post-order = {9, 1, 2, 12, 7, 5, 3, 11, 4, 8}

1) We first take a look at the post-order sequence to find the root of the binary tree. We know that since the post-order method prints out the root last, we can easily pinpoint the root. In this case, the root is 8.

```
        8
```

2) Now we locate the root in the in-order sequence to find the left subtree, and the right subtree. Since the in-order method prints out the left subtree first, then the root, and finally the right subtree, this means that all of the numbers before the root is the left subtree, and the numbers after the root is the right subtree.

In-Order Left = {9, 5, 1, 7, 2, 12}
In-Order Right = {4, 3, 11}

3) With this information, we can get a quick view on how the tree will look like.

```
                        8
                /               \
        {9, 5, 1, 7, 2, 12}      {4, 3, 11}
```

3) Now we use these new sequences and go back to post-order to determine its left and right subtree. But before that, we can count the number of elements in both in-order sequences to be sure the sequences are consistent. In-order left has 6 elements and in-order right has 3 elements.

4) Now knowing the number of elements, since post-order prints out the left subtree first before the right subtree, this means the first 6 elements is the left subtree, and the next 3 elements are its right subtree.

Post-Order Left = {9, 1, 2, 12, 7, 5}
Post-Order Right = {3, 11, 4}

5) Now we repeat mainly steps 2-4 to find each root node of the subtrees. Starting off with the right tree since it has 3 elements, we get a root node of 4 in post-order right.

```
                        8
                          \
                            4
```

6) We go to our in-order right sequence to locate 4. We get
In-Order Left(2) = {}
In-Order Right(2) = {3, 11}

The node 4 will have no left child and will have a right child.

7) Going back to post-order, we have a root of 11 for the right child of 4.

```
    8
     \
      4
       \
        11
```

8) We go back to the in-order sequence and locate 11. We get
In-Order Left(3) = {3}
In-Order Right(3) = {}

9) So 11 will have a left child of 3 and no right child.

```
    8
     \
      4
       \
        11
       /
      3
```

10) Basically, the left subtree follows the same logic as steps 2-4. The root node of post-order left is 5. We get
In-Order Left(2) = {9}
In-Order Right(2) = {1, 7, 2, 12}

```
      8
     / \
    5   4
         \
          11
         /
        3
```

Note that 5 will have a left child of {9} and right child of either {1, 7, 2, 12}

11) Going back to post-order
Post-Order Left(2) = {9}
Post-Order Right(2) = {1, 2, 12, 7}
Getting the less tree out of the way first, the root node is 9.

12) To in-order Left(2), we get
In-Order Left(3) = {}

In-Order Right(3) = {}
So 9 is a leaf node.

```
        8
      /   \
    5       4
  /           \
9               11
               /
             3
```

13) Going back to post-order right(2), the root is 7. We get
In-Order Left(3) = {1}
In-Order Right(3) = {2, 12}

So the left child of 7 is {1}, and the right child is either {2, 12}

```
        8
      /   \
    5       4
  /   \       \
9      7        11
               /
             3
```

14) Our post-order is now
Post-Order Left(3) = {1}
Post-Order Right(3) = {2, 12}

15) Going to the left child of 7 first, we get a root node of 1

```
        8
      /   \
    5       4
  /   \       \
9      7        11
      /        /
    1        3
```

In-Order Left = {}
In-Order Right = {}
So the node of 1 is a leaf node

16) Going back to right child of 7, we get a root node of 12 since
Post-Order Right(3) = {2, 12}

```
          8
        /   \
      5       4
    /  \        \
  9      7       11
       /  \     /
      1    12  3
```

17) Going back to in-order, we get
In Order Left(4) = {2}
In Order Right(4) = {}

So root 12 will have a left child of {2}, and no right children since right child is {}

18) Going back to post-order, we get
Post Order Left(4) = {2}
Post Order Right(4) = {}

We now have a root node of 2 as a left child of 12

```
          8
        /   \
      5       4
    /  \        \
  9      7       11
       /  \     /
      1    12  3
           /
          2
```

19) Going back to in-Order, we get
In-Order Left(5) = {}
In-Order Right(5) = {}
So 2 is a leaf node

20) The sequence is now empty so we are done.
The Binary Tree for this in-order and post-order sequence is:

```
          8
        /   \
      5       4
    /  \        \
  9      7       11
       /  \     /
      1    12  3
           /
          2
```