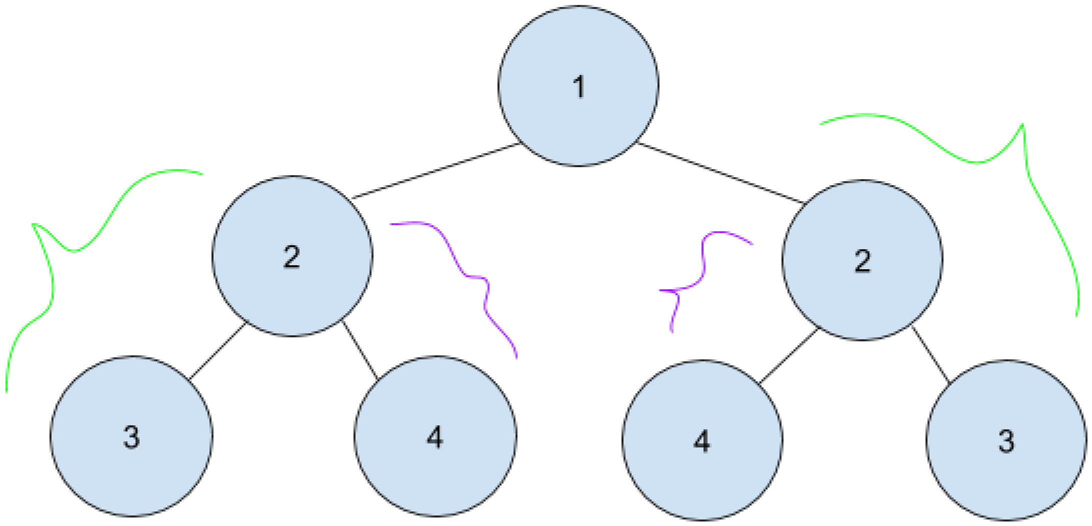
*Symmetric BST Walkthrough*
*Question from Week 10 Mentoring*

```
def is_symmetric(t):
     def helper(t1, t2):
          if t1 is BST.empty and t2 is BST.empty:
               return True
          else if t1 is BST.empty or t2 is BST.empty:
               return False
          else:
               return t1.root == t2.root and
               helper(t1.left, t2.right) and
               helper(t1.right, t2.left)
     return t is BST.empty or helper(t.left, t.right)
```

Approach:

- (Since this method is for a BST, every tree can only have 2 branches, one on the left and one on the right.) What does it mean for a tree to be symmetric? Both its left and right branches are mirror images of each other. Let's say we have a tree that looks like this:

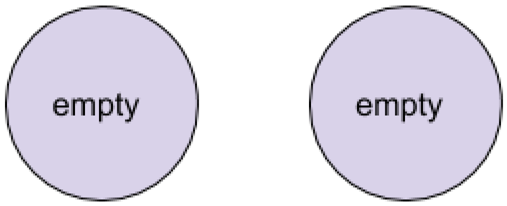


- That is, the left branch of the left branch of a tree is the same as the right branch of the right branch of a tree and the right branch of the left branch of a tree is the same as the left branch of the right branch of a tree.
    - Aka: t.left.left and t.right.right must be symmetric and t.left.right and t.right.left must be symmetric.

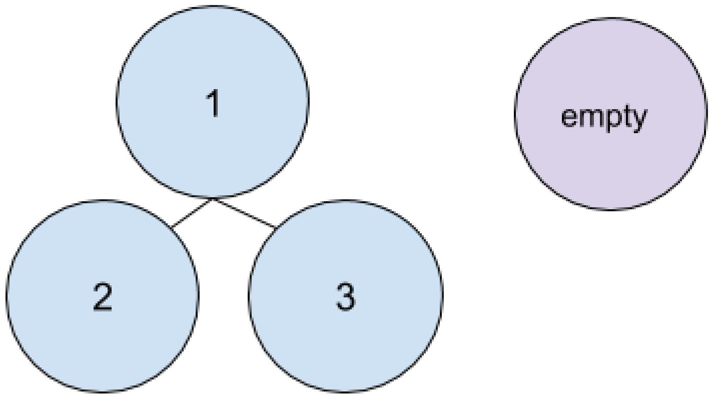
- Since it looks like we are comparing two trees to determine if a tree is symmetric, it might be easier to approach this problem with a helper function that takes in two trees and returns if they are symmetric.

```
def symmetric(t):
     def helper(t1, t2):
```

- Let's think of the base cases for our helper function. If we pass in two empty trees, they are still symmetric.



- However, if we have one empty tree one that isn't empty, they are not symmetric.



```
def symmetric(t):
     def helper(t1, t2):
          if t1 is BST.empty and t2 is BST.empty:
               return True
          else if t1 is BST.empty or t2 is BST.empty:
               return False
```

- If both trees have elements, we need to check that their roots are the same and then check that t1.left.left is symmetric with t2.right.right and t1.left.right is symmetric with t2.right.left. (We came up with this in one of the first steps above!) We can recursively call our helper function here.

```
def symmetric(t):
     def helper(t1, t2):
          if t1 is BST.empty and t2 is BST.empty:
               return True
          else if t1 is BST.empty or t2 is BST.empty:
```

```
                        return False
                else:
                        return t1.root == t2.root and
                        helper(t1.left, t2.right) and
                        helper(t1.right, t2.left)
```

- Now that we have completed our helper function, let’s write the return statement for our first function. A tree is symmetric if it’s empty or if it’s branches are mirror images.

```
def symmetric(t):
        def helper(t1, t2):
                if t1 is BST.empty and t2 is BST.empty:
                        return True
                else if t1 is BST.empty or t2 is BST.empty:
                        return False
                else:
                        return t1.root == t2.root and
                        helper(t1.left, t2.right) and
                        helper(t1.right, t2.left)
return t is BST.empty or helper(t.left, t.right)
```

The hardest part of this question was probably figuring out that we should use a helper function, but once we got that part, I think the question becomes a lot easier!