CS 314 Exam Review — Get Grandparent — **Solution**

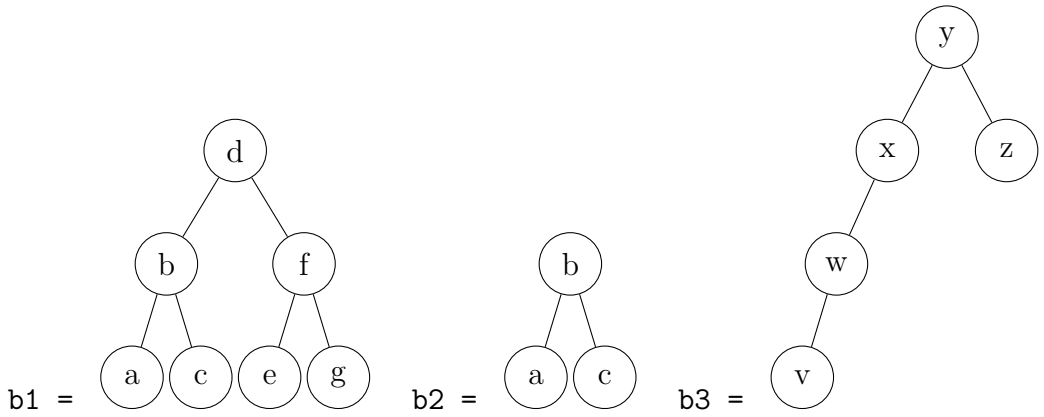
Binary Trees

Write an instance method for a BinarySearchTree which, given an element returns that element's grandparent in the tree. If the given element is not in the tree, return `null`. If the element does not have a grandparent (i.e. the element is not deep enough), return the root element.

Complete the following method.

```
// Returns the grandparent of 'data'
// pre: data != null
// post: This tree shall not be altered by this operation
public E getGrandparent(E data){
```

Here are some sample calls to `getGrandparent`:




```
b1.getGrandparent('a') → 'd'      b1.getGrandparent('g') → 'd'
b2.getGrandparent('b') → 'b'      b2.getGrandparent('c') → 'b'
b3.getGrandparent('a') → null     b3.getGrandparent('v') → 'x'
```

You may use the following BinaryTree implementation

```
public class BinarySearchTree<E extends Comparable<? super E>>{
  BSTNode<E> root;
  int size;

  //Nested node class
  private static class BSTNode<E>{
    BNode<E> left, right;
    E data;
  }
}
```

Do not create any new data structures or use any other Java classes or methods.

```
// Returns the grandparent of 'data'
// pre: data != null
// post: This tree shall not be altered by this operation
public E getGrandparent(E data){
    // Start parent and grandparent as root in case tgt isn't deep
    // enough to have a grandparent
    return helper(tgt, root, root, root);
}

public E helper(E tgt, BSTNode<E> curr, BSTNode<E> parent, BSTNode<E> gp){
    // Fell off the tree, tgt is not in the tree
    if(curr == null)
        return null;

    int compare = tgt.compareTo(curr.data);
    if(compare == 0){
        // Found the target, return the grandparent's data
        return gp.data;
    } else if (compare < 0){
        return helper(tgt, curr.left, curr, parent);
    } else {
        return helper(tgt, curr.right, curr, parent);
    }
}
```