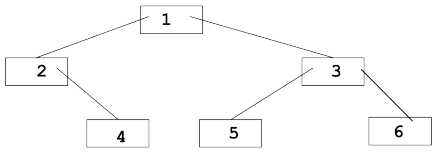
C Sc 127B Practice Test 3 ANSWERS SL _____ Name _____ 100pts

1. Use the following binary tree to write out each of the three traversals indicated below.



Preorder traversal _ 1 2 4 3 5 6

Inorder traversal _ 2 4 1 5 3 6

Postorder traversal _ 4 2 5 6 3 1

2. Is this a Binary Search Tree assuming only String keys are shown? (yes or no)? _No (must be unique)

3. Is this a Binary Search Tree assuming only String keys are shown? (yes or no)? _No (reverse order)
Actually it could be if we defined the order property to be All nodes to the right are less than, to the left greater than.

4. What potential benefit is there to using a binary search tree rather than an array to store the same collection of key/value mappings of a Map?

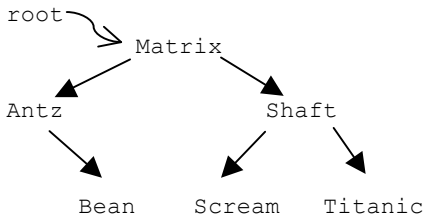
Better runtimes for put, get, and remove. With a BST, they are O(log n), with an array some are O(n)

5. What would be required of the keys for those mappings to be stored in a binary search tree?

They are Comparable, that is the element implements the Comparable interface so we can see if a key is less than or greater than another

6. Draw a picture ....

```
aBST.put("Matrix");
aBST.put("Antz);
aBST.put("Bean");
aBST.put("Shaft");
aBST.put("Scream");
aBST.put("Titanic");
```



7.

```
public int nodesAtLevel(int level) {
    return atDepth(root, level);
}

 private int atDepth(TreeNode t, int level) {
   if (t == null)
       return 0;
   else if (level == 0)
       return 1;
   else
       return atDepth(t.left, level - 1) + atDepth(t.right, level - 1);
}
```

```
8.public List<E> elementsGreaterThan(E element) {
     ArrayList<E> list = new ArrayList<E>();
     elementsGreaterThan(root, element, list);
     return list;
  }
```

```
private void elementsGreaterThan(TreeNode t, E element, ArrayList<E> list) {
  if (t != null) {
    elementsGreaterThan(t.left, element, list);
    if (t.data.compareTo(element) > 0)
      list.add(t.data);
    elementsGreaterThan(t.right, element, list);
  }
}
```

9. Two answers (others possible)

Assume you have public int nodesAtLevel(int level) {

```
public boolean bottomLevelFilled() {
  int level = 0;
  int shouldHave = (int) Math.pow(2, level); // 1 at first
  int nodesFound = nodesAtLevel(level);
  while (nodesFound > 0) {
    if (nodesFound < shouldHave)
      return false;
    ++level;
    shouldHave = (int) Math.pow(2, level);
    nodesFound = nodesAtLevel(level);
  }
  return true; // all levels had the max value
}
```

Assume you do NOT have public int nodesAtLevel(int level) {

```
public boolean bottomLevelFilled() {
  int levels = height(root);
  int shouldHave = (int) Math.pow(2, levels + 1) - 1;
  int totalNumberOfNodes = size(root);
  return shouldHave == totalNumberOfNodes;
}

private int size(TreeNode t) {
  if (t == null)
    return 0;
  else
    return 1 + size(t.left) + size(t.right);
}

private int height(TreeNode t) {
  if (t == null)
    return -1;
  else
    return 1 + Math.max(height(t.left), height(t.right));
}
```

10. Complete the output that would be generated by this code

```
-5 -5 12 23
===
true
false
-5
23
3
```

11.

```
null
null
M
null
3
B
C
D
null
```

12

Key	Value: ArrayList<Character>()
" a"	[b, b, t]
"ab"	[b, l]
"ba"	[ ] one space
"a "	[a]
"bl"	[e]
"le"	[ ] one space
"e "	[a]
"at"	[.]

13. What is your favorite color __Blue or Pink, or Green, or Whatever__ (4pts)

Do not leave the last question blank, any answer will be worth 4 pts