Section #6

Ashley Taylor and Shreya Shankar (Based on handouts by previous CS106B instructors and TAs)

Trees

1. Size

```
int size(BinaryTreeNode* node) {
    if (node == nullptr) {
        return 0;
    } else {
        return 1 + size(node->left) + size(node->right);
    }
}
```

2. Is Balanced

```
int height(BinaryTreeNode* node);

bool isBalanced(BinaryTreeNode* node) {
    if (node == nullptr) {
        return true;
    } else if (!isBalanced(node->left) || !isBalanced(node->right)) {
        return false;
    } else {
        int leftHeight  = height(node->left);
        int rightHeight = height(node->right);
        return abs(leftHeight - rightHeight) <= 1;
    }
}

int height(BinaryTreeNode* node) {
    if (node == nullptr) {
        return 0;
    } else {
        return 1 + max(height(node->left), height(node->right));
    }
}
```

3. Is Binary Search Tree?

```
bool isBSTHelper(BinaryTreeNode* node, BinaryTreeNode*& prev) {
    if (node == nullptr) {
```

```
        return true;
    } else if (!isBSTHelper(node->left, prev) || (prev && node->data <= prev->data)) {
        return false;
    } else {
        prev = node;
        return isBSTHelper(node->right, prev);
    }
}

bool isBST(BinaryTreeNode* node) {
    BinaryTreeNode* prev = nullptr;
    return isBSTHelper(node, prev);
}
```

4. Follows Min Property?

```
bool followsMinProperty(TreeNode *root) {
    if (root == NULL) {
        return true;
    }
    if (root->left != NULL && root->left->data < root->data) {
        return false;
    }
    if (root->right != NULL && root->right->data < root->data) {
        return false;
    }
    return followsMinProperty(root->left) && followsMinProperty(root->right);
}
```

6. Heap Add/Remove

after all enqueues: {e:9, h:16, f:34, c:40, j:22, a:68, g:94, b:77, i:47, d:70, l:77}

after two dequeues: {j:22, c:40, f:34, i:47, d:70, a:68, g:94, b:77, l:77}