Name: _____

Today's Date: _____

Today's Goals

- Implement insert in two different ways
- Motivate and define tree rotations
- Explain how randomized trees work

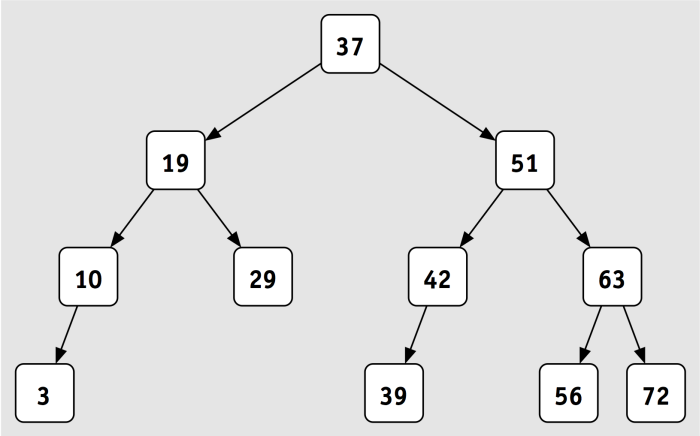
Today's Question(s)

Every function we write for a Tree will be a wrapper to a helper function that works on Nodes and that is _____

Lingering Questions

Reminder: Binary Search Trees (BSTs)

- Each node has at most 2 children
- All nodes in left subtree are less than parent
- All nodes in right subtree are greater than parent



Reminder: Tree Encoding

```
class StringTreeSet {
public:
  ...
private:
  struct Node {
    string      value_;
    Node* left_;
    Node* right_;
  };
  Node* root_;
};
```

Reminder: Tree Lookup

```
bool lookupNode(const Node* node, const string& key) {
    if (node == nullptr) {
        return false;
    } else if (key < node->value_) {
        return lookupNode(node->left_, key);
    } else if (node->value_ < key) {
        return lookupNode(node->right_, key);
    } else
        return true;
    }
```

Insertion

```
void StringTreeSet::nodeInsert(Node*& node, const string&
    if (node == nullptr) {

    } else if (key < node->value_) {

    } else if (node->value_ < key) {

    } else {
        return; // Duplicate is undefined behavior
    }
}
```

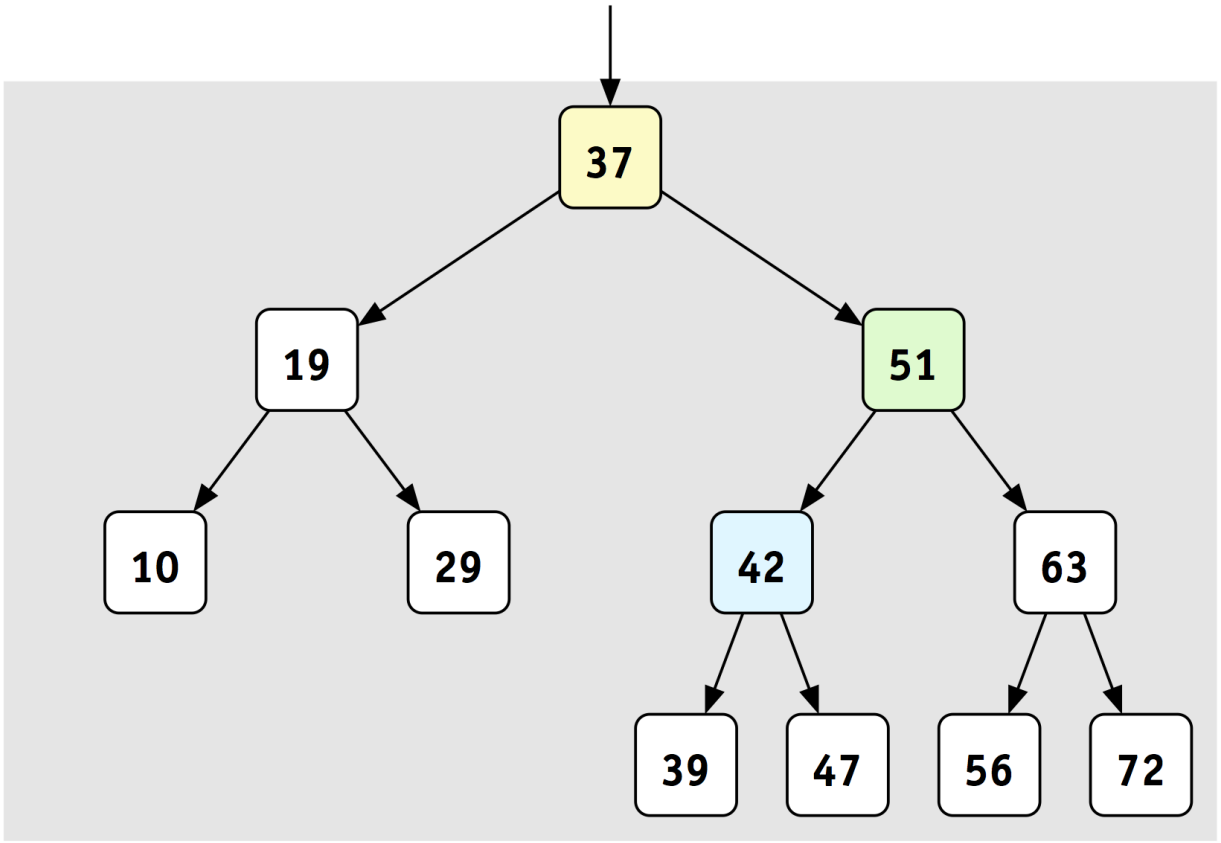
Practice: Create a tree from one of these

- D B F C G E A
- A B C D E F G
- D C B A E F G

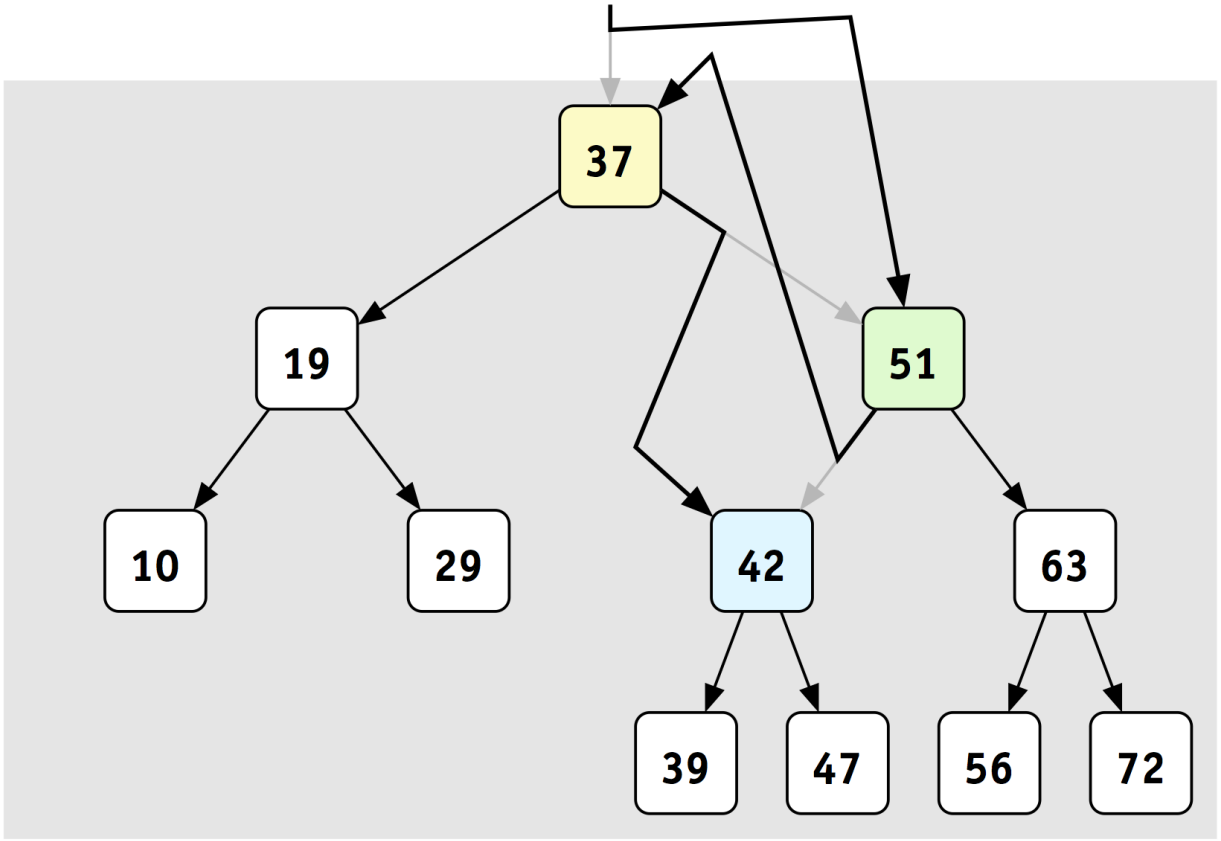
Rearranging Trees

Rearrange the tree to make 51 the root (in constant time)

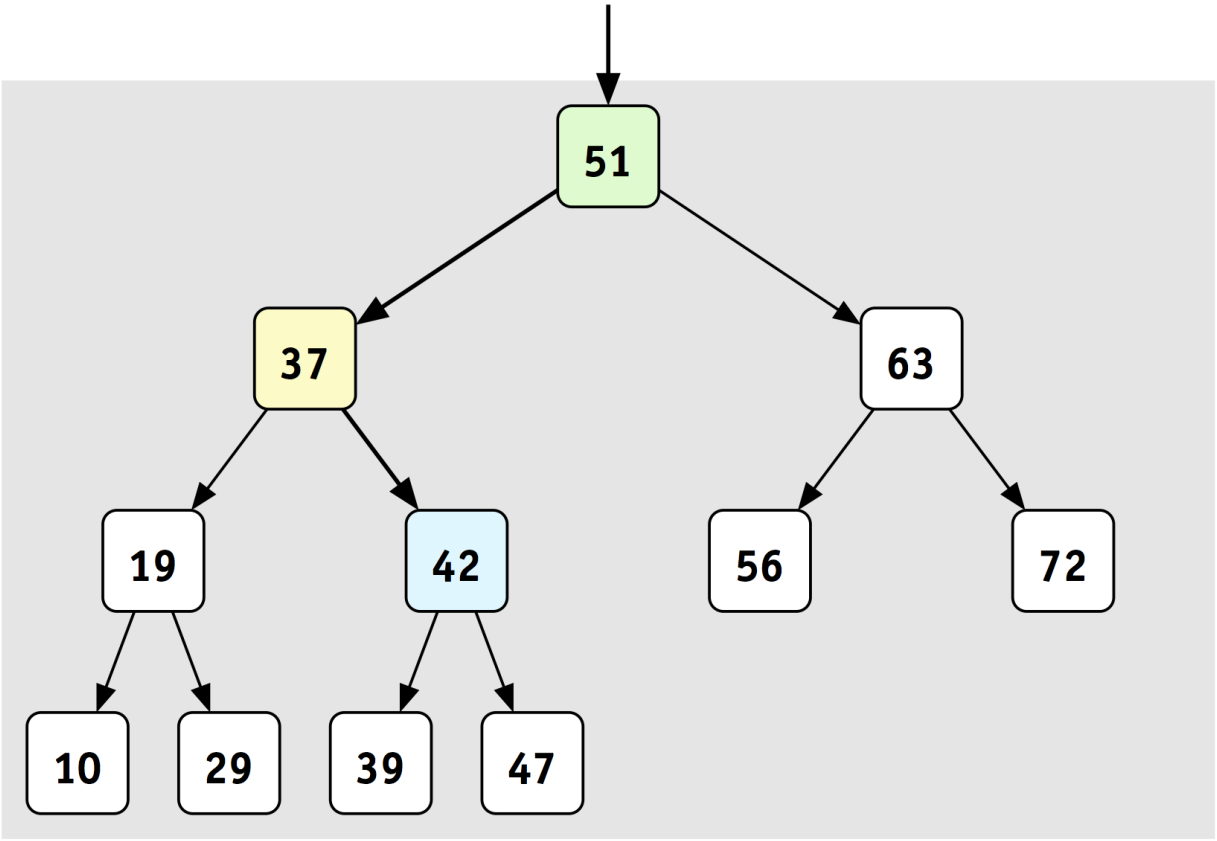
Rotations



Rotations



Rotations



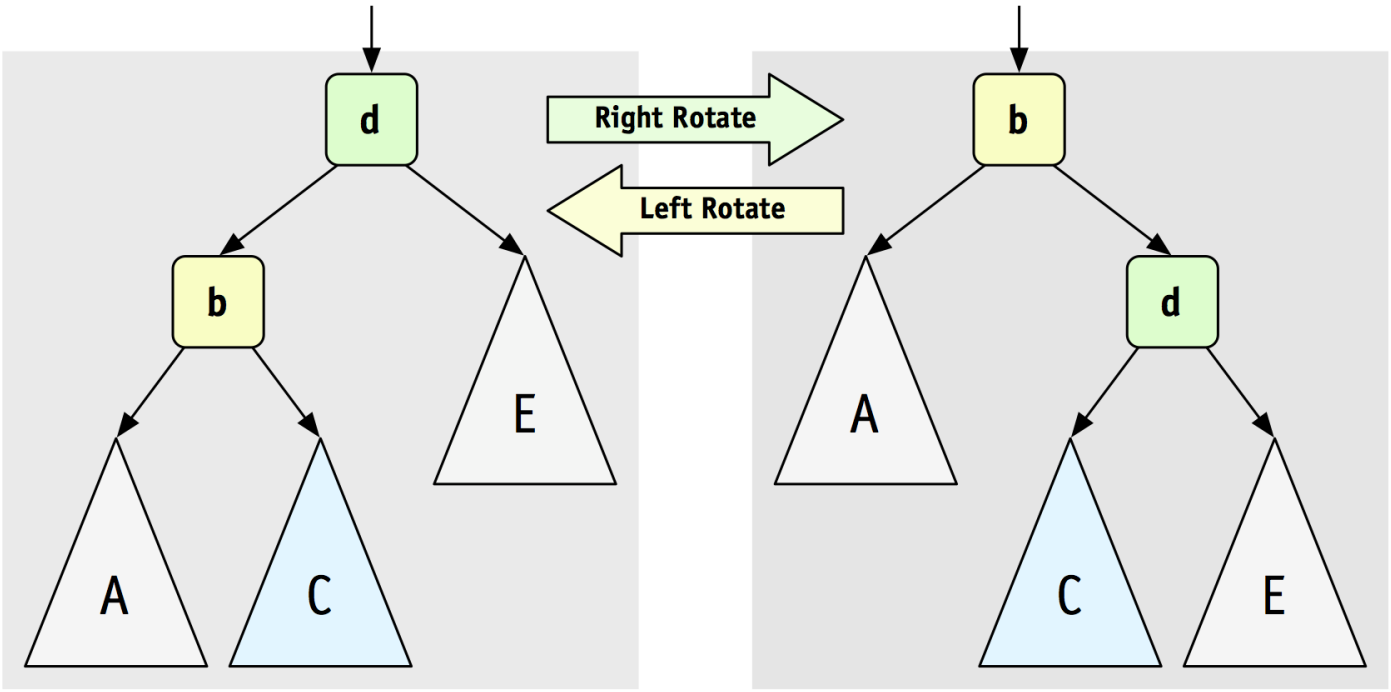
Rotation Code

```
void rotateRight(Node*& top) {




}
```

Rotations



Class Exercise

Write insertAtRoot, which

- inserts a value at a leaf, then
- "bubble the value up" to the root

Hint: *use rotations*

insertAtRoot

```
void insertNodeAtRoot(Node*& node,
                      const string& key) {




}
```

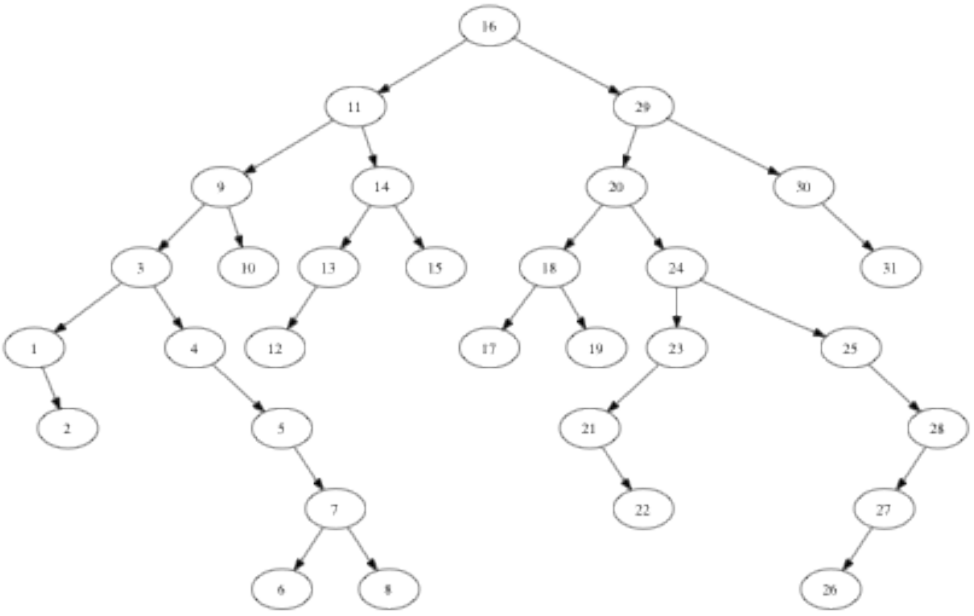
Does this fix our stick problem?

- A B C D E F G

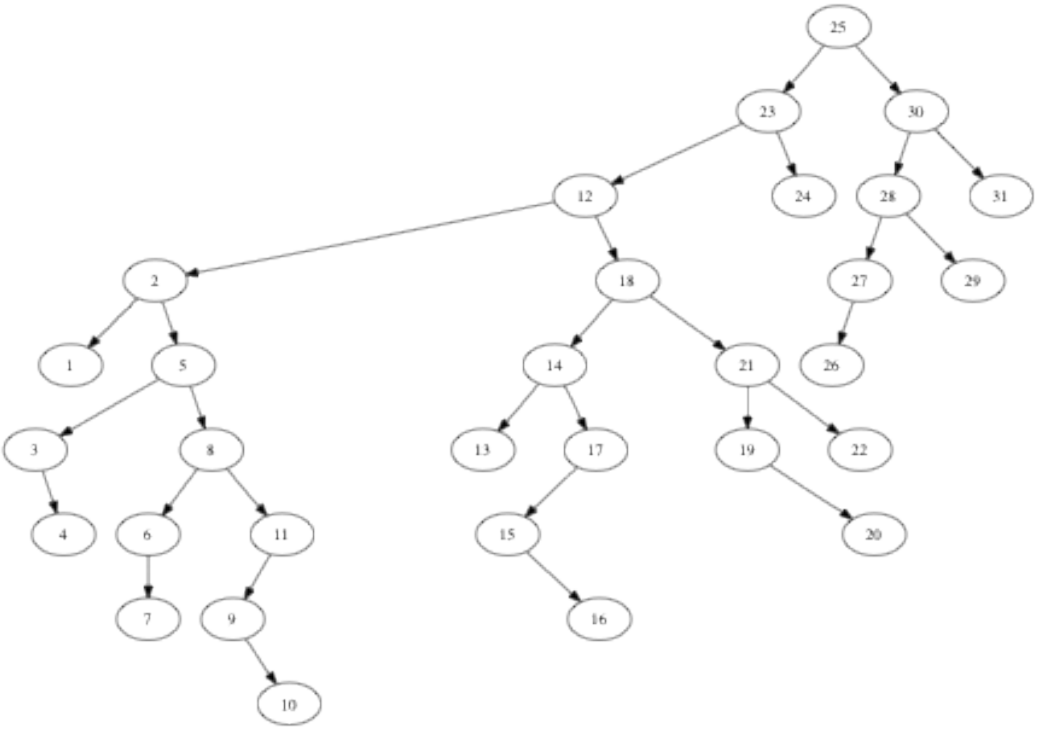
Random Trees

What do they look like?

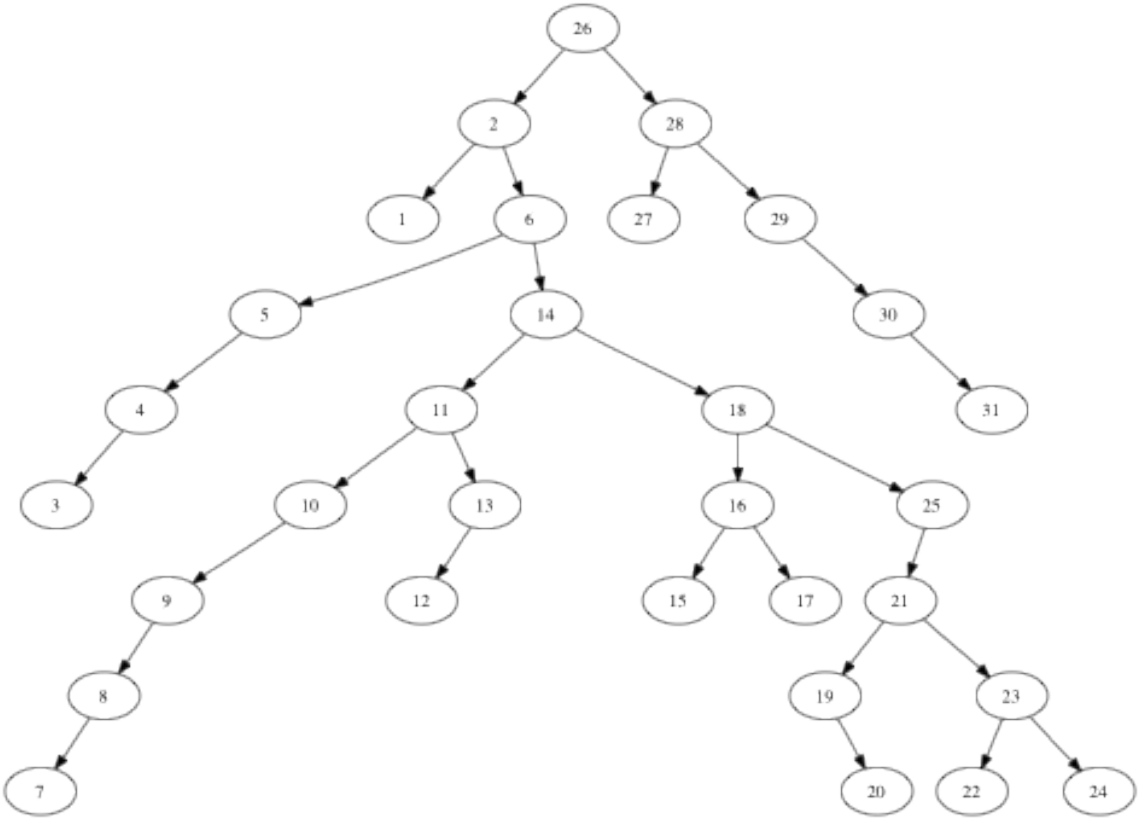
What are their properties?



Random Trees



Random Trees



Unsuccessful Search

Compared to perfection:

$$\lim_{n\to\infty} \frac{2H_{n+1} - 2}{\log_2(n+1)} = 2\log 2 \approx 1.38629$$

Random Trees Are Good!

Observe that *if keys come in in random order*, the tree will tend to be reasonably balanced (about 38% worse than a perfectly balanced tree).

Randomized Trees

Simulate random insertion by

–

–

But how often should we do each?