AVL Tree Insertion

Start out by using a regular binary search tree insertion. Set the balance factor of the newly inserted node to zero. Call insertionFixUp, passing a pointer to the newly inserted node.

```
function insertionFixUp(x) //x is the newly inserted node
{
    loop
        { // favor right over left
            if (x is root)
                exit the loop
            else if (sibling is favorite child of parent)
                { // favor right over left
                    set favorite of parent to null
                    exit the loop
                }
            else if (x is favorite child of parent)
                { // x must have a child
                    c = favorite child of x
                    if (c, x, and parent not linear)
                        { // favor right over left
                            rotate c to x
                            rotate c to parent
                        }
                    else
                        rotate x to parent
                    exit the loop
                }
            else
                { // favor right over left
                    set favorite of parent to x
                }
            x = parent
        }
}
```

Note that in this pseudocode, there are no references to leftness and rightness. This issue is deferred to the helper functions. For example, the getting the sibling of a node \( c \) with parent \( p \) could be implemented as:

```
function getSibling(c, p)
{
    if (p.left == c)
        return p.right;
    else
        return p.left;
}
```

Likewise, asking if a child is the favorite of the parent could be implemented as:

```
function isFavorite(c, p)
{
    return (p.left == c && p.balance == 1)
        || (p.right == c && p.balance == -1);
}
```

Next: *Deleting from AVL trees*