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.

```plaintext
function insertionFixUp(x) //x is the newly inserted node
{
    loop
    {
        if (x is root)
            exit the loop
        else if (sibling is favorite child of parent)
            { //x must have a child
c = favorite child of x
            if (c, x, and parent not linear)
                {
                    if (c has a sibling)
                        set the favorite child of x to that sibling
                    if (the grandparent's favorite child is the parent)
                        set the grandparent's favorite child to c
                    set favorite of c to null
                    rotate c to x
                    rotate c to parent
                }
            else
                { //x must have a child
                    if (an existing grandparent's favorite is the parent)
                        set the grandparent's favorite child to x
                    rotate x to parent
                }
            exit the loop
        }
    }
    else
    {
        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:

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

Next: Deleting from AVL trees