Start out by using a regular binary search tree insertion. Set the favorite of the newly inserted node to null. Call `insertionFixUp`, passing a pointer to the newly inserted node.

```plaintext
function insertionFixUp(x) //x is the newly inserted node
{
    loop
    {
        // set parent, grandparent, sibling here
        // these are the "originals"
        if (x is root)
            exit the loop
        else if (sibling is favorite child of parent)
            {
                set the favorite of the parent to none
                exit the loop
            }
        else if (x is favorite child of parent)
            {
                //x must have a child
                c = favorite child of x
                cf = favorite child of c
                cs = sibling of c
                if (c, x, and parent not linear)
                    {
                        if (the parent is the favorite of the the grandparent)
                            set the favorite of the grandparent to c
                        set the favorite of c to none
                        set the favorite of x to none
                        set the favorite of parent to none
                        rotate c to x
                        rotate c to parent
                        //cf balances its new parent
                        //whoever didn't end up with cf is unbalanced
                        if (cf exists and cf's parent != x)
                            set the favorite of x to cs
                        if (cf exists and cf's parent != original parent)
                            set the favorite of the original parent to the original sibling
                    }
                else
                    {
                        if (an existing grandparent's favorite is the parent)
                            set the favorite of the grandparent to x
                        set the favorite of x to none
                        set the favorite of parent to none
                        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:

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

Next: Deleting from AVL trees