RED BLACK TREE: https://en.wikipedia.org/wiki/Red%E2%80%93black_tree

- Delete a *key-value* pair
  - Step 1: delete as a BST
    - Search the *key*, recursively or iteratively
    - If not found, then deletion failed and return `null`
    - Otherwise (*node* found):
      1. If *node* has two non-`NIL` children, then the next larger node is the left-most node in *node.rChild*'s subtree, and this *larger* has at least a `NIL` child. Swap *node* with *larger*
      2. The *node* now has at least one `NIL` child: set *node* to `NIL`'s sibling (this sibling may also be `NIL`)
      3. If *node* is not *root* (*parent* not `null`), then link *child* to *parent*; otherwise:
         - If *child* is `NIL`, then empty the tree
         - Otherwise, set *root* to *child*
      4. If *node*'s color was black:
         - If *child*'s color is red, simply set *child*'s color to black
         - Otherwise, fix *child*'s color by Step 2
      5. Return *node.value*

Important note: the Wikipedia RBT removal has errors in both description and source code, so please read with caution.

- Step 2: balance the tree – `fixDelColor(node)`
  - Case 1, *node* is *root*: done (it's already black)
  - Case 2, *sibling* of *node* is red:
    - Set *parent*'s color to red, and *sibling*'s color to black
    - Rotate left/right on *sibling* if *node* is a lChild/rChild
    - Update *sibling* to *node*'s new sibling
  - Case 3, *parent*, *sibling*, *sibling.lChild* and *sibling.rChild* are all black: set *sibling*'s color to red, and fix *parent*'s color – invoke `fixDelColor(parent)`
  - Case 4, *parent* is red, *sibling*, *sibling.lChild* and *sibling.rChild* are all black: simply set *sibling* to red and *parent* to black
  - Case 5, sibling's color is black: set *sibling* to red, and:
    - *node* is a lChild, *sibling.lChild* is red, and *sibling.rChild* is black: set *sibling.lChild* to black, and rotate right on *sibling.lChild*
    - *node* is a rChild, *sibling.lChild* is black, and *siblng.rChild* is red: set *sibling.rChild* to black, and rotate left on *sibling.rChild*
  - Case 6: set *sibling*'s color to *parent*'s color, and then *parent*'s color to black
    - If *node* is a lChild, then set *sibling.rChild*'s color to black, and rotate left on *sibling*
    - If *node* is a rChild, then set *sibling.lChild*'s color to black, and rotate right on *sibling*