BST Delete Logic

Precondition: We're given a pointer to the node to delete, call it pointer, and that pointer is located in the node's parent.

Four cases

1. Node to delete is a leaf
   set pointer to null
   delete node

2. Node to delete has only a left subtree
   pointer <- delete's left subtree
   delete node

3. Node to delete has only a right subtree
   pointer <- delete's right subtree
   delete node

4. Node to delete has both a left and a right subtree

   (1) adjust key - find node with highest key that's not > node to delete
      start at node to delete (pointer)
      go to its left child (we know it has one)
      go to its rightmost child (it may already be the rightmost)
      take that key and put into the node to delete

   (2) adjust subtree
      if the left child was already the rightmost
         node to delete's left <- left child's left
      if the left child had a rightmost
         rightmost's parent's right <- rightmost's left
      delete rightmost