```
public boolean removeKey(int k) {
      // Search for k
      int k1 = k;
     BSTNode<T> p = root;
      BSTNode<T> q = null; // Parent of p
      while (p != null) {

         if (k1 < p.key) {
            q =p;
            p = p.left;
         } else if (k1 > p.key) {
            q = p;
            p = p.right;
         } else { // Found the key

            // Check the three cases
            if ((p.left != null) && (p.right != null)) { // Case 3: two
                                                         // children
               // Search for the min in the right subtree
               BSTNode<T> min = p.right;
               q = p;
               while (min.left != null) {
                  q = min;
                  min = min.left;
               }
               p.key = min.key;
               p.data = min.data;
               k1 = min.key;
               p = min;
               // Now fall back to either case 1 or 2
            }

            // The subtree rooted at p will change here
            if (p.left != null) { // One child
               p = p.left;
            } else { // One or no children
               p = p.right;
            }

            if (q == null) { // No parent for p, root must change
               root = p;
            } else {
               if (k1 < q.key) {
                  q.left = p;
               } else {
                  q.right = p;
               }
            }
            current = root;
            return true;

         }
      }

      return false; // Not found
   }
```