1. My code from lecture 19 and lab 10 had a bug in it!

```
void IntAVLTree::insert(TreeNode *&nodePtr, TreeNode *&newNode, bool & taller) {

      bool tallerLeftSubtree, tallerRightSubtree;

      taller = false;   // BUG FIX FOR LECTURE 19 AND LAB 10 CODE<<<<<<<<<<<<<<<<<<<<<<<

      if (nodePtr == NULL) {
            nodePtr = newNode;            // Insert the node as a leaf
            taller = true;
      } else if (newNode->value < nodePtr->value) {  // insert into the left subtree

            // ADD CODE HERE FOR PART A OF LAB 10

      } else {  // insert into the right subtree
            insert(nodePtr->right, newNode, tallerRightSubtree);
            if (tallerRightSubtree) {
                  if (nodePtr->balance == TR) {
                        // Need rotation(s) to restore AVL height-balance property
                        rebalanceRightSubtree(nodePtr);
                        taller = false;
                  } else if (nodePtr->balance == EQ) {
                        nodePtr->balance = TR;
                        taller = true;
                  } else {
                        nodePtr->balance = EQ;
                        taller = false;
                  } // end if

            } // end if (tallerRightSubtree)

      } // end if
}
```

a) What bug does the above line fix?

2. Hashing Motivation and Terminology:

a) Sequential search of an array or linked list follows the same search pattern for any given target value being searched for, i.e., scans the array from one end to the other, or until the target is found.

If n is the number of items being searched, what is the average and worst case theta notation for a search?

average case $\Theta($ $)$

worst case $\Theta($ $)$

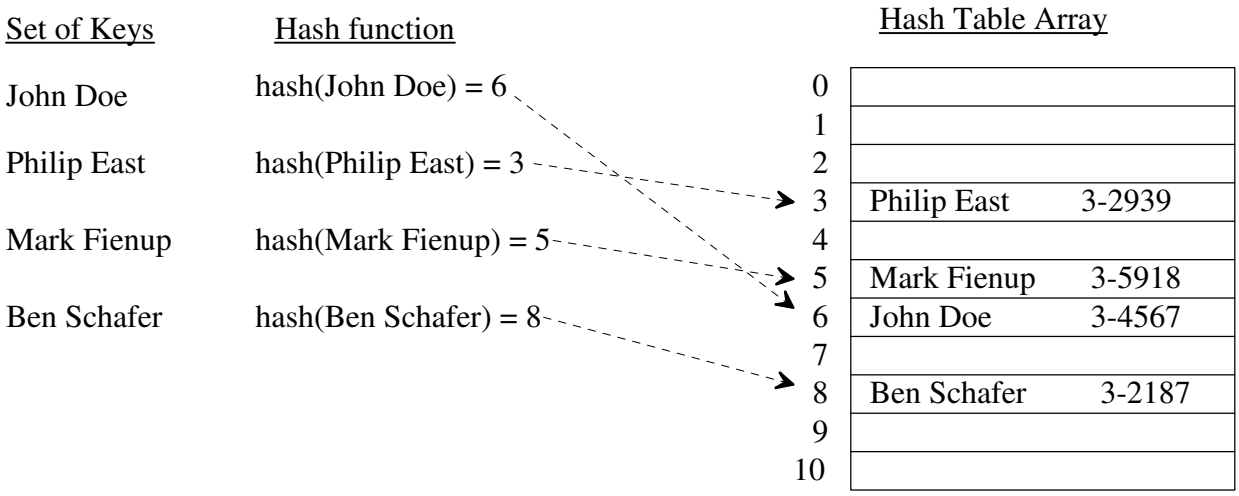
b) Similarly, binary search of a sorted array or AVL tree always uses a fixed search strategy for any given target value. For example, binary search always compares the target value with the middle element of the remaining portion of the array needing to be searched.

If n is the number of items being searched, what is the average and worst case theta notation for a search?

average case $\Theta($ $)$

worst case $\Theta($ $)$

Hashing tries to achieve average constant time (i.e., $O(1)$) searching by using the target's value to calculate where in the array (called the *hash table*) it should be located, i.e., each target value gets its own search pattern. The translation of the target value to an array index (called the target's *home address*) is the job of the *hash function*. A *perfect hash function* would take your set of target values and map each to a unique array index.



a) If n is the number of items being searched and we had a perfect hash function, what is the average and worst case theta notation for a search?

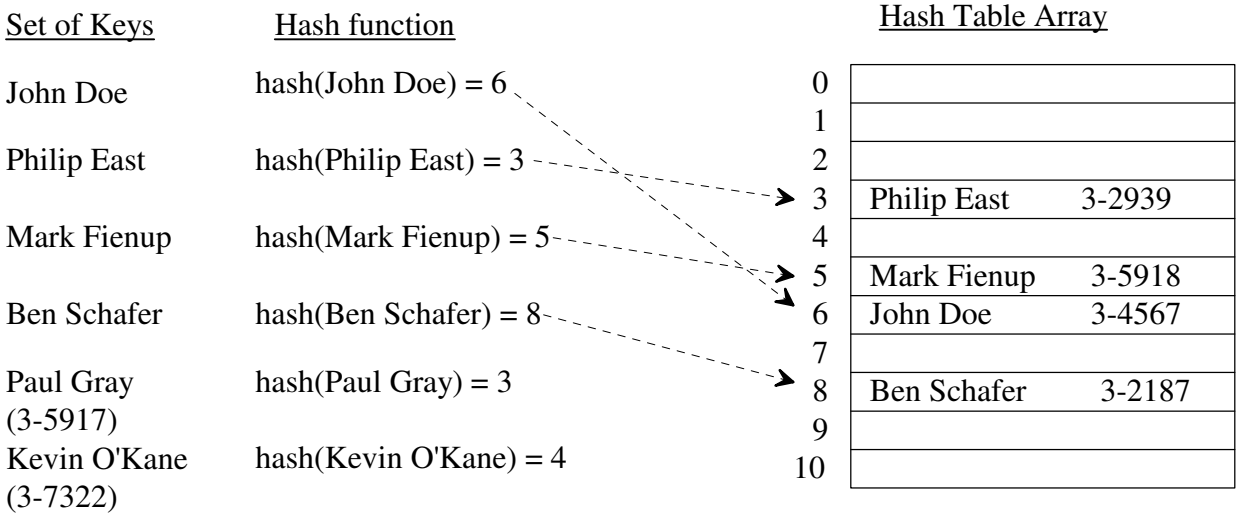
average case $\Theta($ $)$

worst case $\Theta($ $)$

3. Unfortunately, perfect hash functions are a rarity, so in general two or more target values might get mapped to the same hash-table index, called a *collision*.

Collisions are handled by two approaches:

- *chaining, closed-address*, or *external chaining*: all target values hashed to the same home address are stored in a data structure (called a *bucket*) at that index (typically a linked list, but a BST or AVL-tree could also be used). Thus, the hash table is a array of linked list (or whatever data structure is being used for the buckets)
- *open-address* with some *rehashing* strategy: Each hash table home address holds at most one target value. The first target value hashed to a specify home address is stored there. Later targets getting hashed to that home address get rehashed to a different hash table address. A simple rehashing strategy is linear probing where the hash table is scanned circularly from the home address until an empty hash table address is found.



a) Assuming open-address with linear probing where would Paul Gray and Kevin O'Kane be placed?