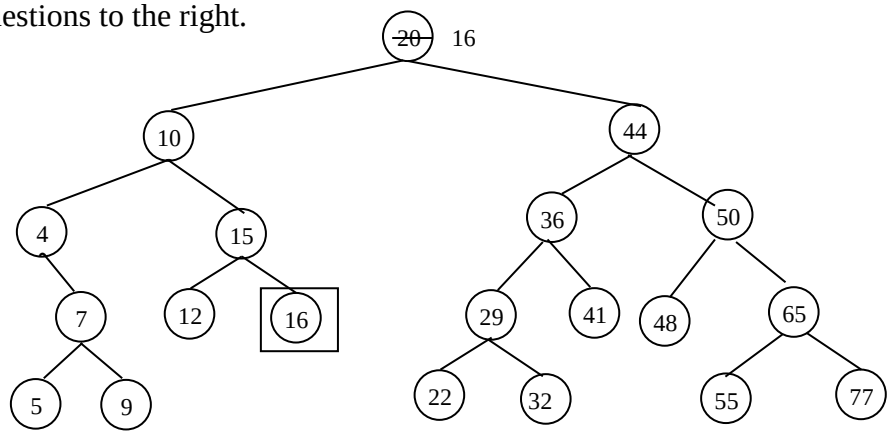
Quiz #4: Binary [Search] Trees, Traversals, Heaps ICS-46 Fall 2016

Name _____

Recall the Academic Integrity statement that you signed. Write all answers clearly on these pages, ensuring your final answers are easily recognizable. The number of points for each problem is clearly marked, for a total of 25 points. I will post my solutions on the web on Friday, off the **Solutions** link, after class.

---

1. (4 pts) Build a binary search tree by adding the following values, in the order specified, to an empty tree: 20, 10, 4, 44, 50, 48, 36, 29, 15, 7, 12, 65, 41, 32, 5, 16, 77, 22, 9, 55. Show this tree below. Answer the questions to the right.



Size = 20

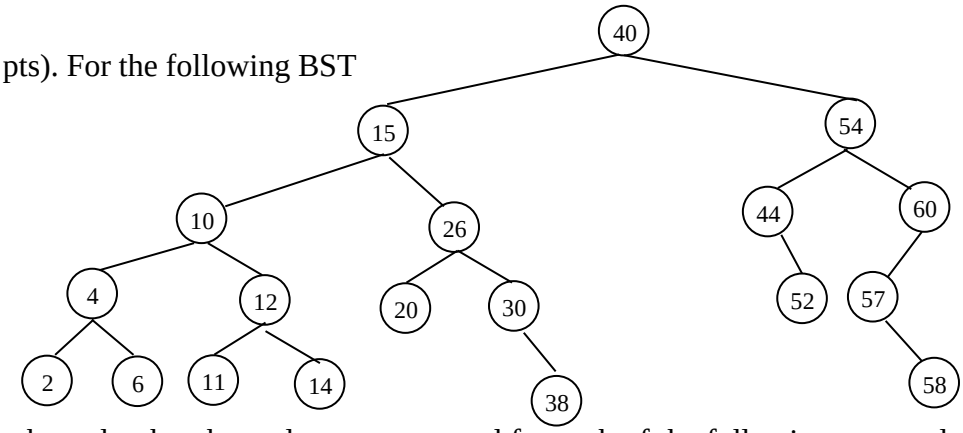
Height = 4

What's the minimum number of nodes to remove to decrease the height? 6

Or, box 22 and put its value at the 20 node

Now, using the picture you drew above, remove the value 20 draw a **box** around node(s) that are actually removed from the tree; cross out/rewrite values inside ant node(s) that change.

2. (4 pts). For the following BST



Show the order that the nodes are processed for each of the following traversals. It is standard (unless told otherwise) to process left subtrees before right ones in all these traversals.

Preorder	40	15	10	4	2	6	12	11	14	26	20	30	38	54	44	52	60	57	58
Inorder	2	4	6	10	11	12	14	15	20	26	30	38	40	44	52	54	57	58	60
Post Order	2	6	4	11	14	12	10	20	38	30	26	15	52	44	58	57	60	54	40
Breadth 1st	40	15	54	10	26	44	60	4	12	20	30	52	57	2	6	11	14	38	58

For the problems,3-4, assume that we have declared the standard template **`TN`** class (see the notes).

3. (4 points) Write the recursive function named **`all_less`** which takes a **`TN<T>*`** (a pointer to any arbitrary binary tree, not necessarily a search tree) and const **`T&`** parameter; it returns whether or not all the values in the tree are less than the **`v`** parameter. Think a bit and write the simplest recursive code. Use one if statement, and declare no local variables. Write your answer on the top of the next page

```
template<class T>
bool all_less (TN<T>* t, const T& v) {
  if (t == nullptr)
    return true;
  else
    return t->value < v  &&  all_less(t->left,v)  &&  all_less(t->right,v);
}
```

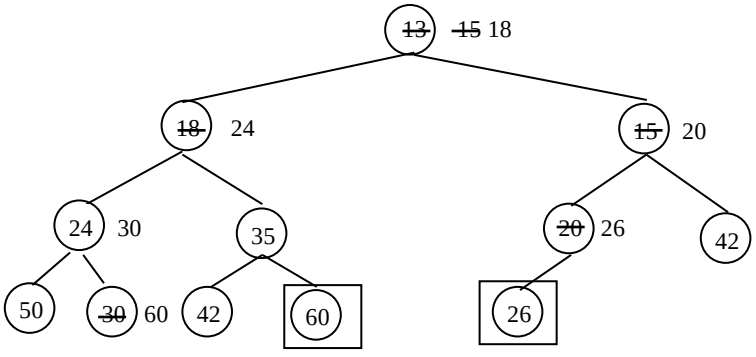
Note the solution is symmetric with regards to both children; we can recur in either order.

4. (5 points) Assume that the function **all_less** written above (and another, **all_greater**) have been written correctly. Write the **recursive** function named **is_BST** which takes a **TN<T>** (any arbitrary binary tree, **not necessarily a search tree**) and returns whether or not that tree satisfies the order property for a binary search tree (so it should return **true** for the tree you constructed in part 1 above). Use one if statement, and declare no local variables.

```
template<class T>
bool is_BST (TN<T>* t) {
  if (t == nullptr)
    return true;
  else
    return all_less(t->left,t->value) && all_greater(t->right,t->value) &&
           is_BST(t->left) && is_BST(t->right);
}
```

Note the solution is symmetric with regards to both children, in regards to both **allLess**/**allGreater**. It must check **isBST** for both it children as well: every node in the tree must check that its value is between all the values in its left and right descendants.

5. (8 pts) (a) Build a Min-Heap by **adding** the following values, **in the order shown**, to an empty Min-Heap (don't just draw a Min-Heap storing these value). Insert, 42, then 26, 13, 50, 30, 20, 42 (again), 18, 24, 35, 60, and insert 15 last. Show it (as a tree) below. Do the work elsewhere and **transcribe your final answer** here, when you are finished. **Don't show intermediate work**, just the final tree.



(b) Fill in the values for the array representation of this final Min-Heap (using the standard mapping).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
13	18	15	24	35	20	42	50	30	42	60	26								

(c) Remove the minimum value **twice**, just in the tree you drew above (not the array): **draw a box** around node(s) that are removed; **cross out/rewrite** changed values inside node(s) as was done in the reading and class.