READ THIS NOW!

- Print your name in the space provided below.
- There are 6 short-answer questions, priced as marked. The maximum score is 100.
- Aside from the allowed one-page fact sheet, this is a closed-book, closed-notes examination.
- No laptops, calculators, cell phones or other electronic devices may be used during this examination.
- You may not discuss this examination with any student who has not taken it.
- Failure to adhere to any of these restrictions is an Honor Code violation.
- When you have finished, sign the pledge at the bottom of this page and turn in the test and your fact sheet.

Name (Last, First) Solution

printed

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

signed

1. Consider the following algorithm:

```
for (i = 1; i <= N; i++) {           // 1 before, 2 each pass, 1 to exit

   for (j = i; j <= N; j++) {        // 1 before, 2 each pass, 1 to exit

      if ( i * j <= N ) {            // 2 each pass
         less = less + 5;            // 2 if done
      }
      else {
         less = less - 1;            // 2 if done
      }
   }
}
```

a) [14 points] Using the rules given in class for counting operations, find a simplified function T(N) that counts the exact number of operations the algorithm would perform.

From the line-by-line analysis above, the complexity function would be:

$$
\begin{aligned}
T(N) &= 1 + \sum_{i=1}^{N}\left(2 + 1 + \sum_{j=1}^{N}(2+2+2) + 1\right) + 1 \\
&= \sum_{i=1}^{N}\left(\sum_{j=i}^{N} 6 + 4\right) + 2 \\
&= \sum_{i=1}^{N}\left(\sum_{j=1}^{N} 6 - \sum_{j=1}^{i-1} 6 + 4\right) + 2 \\
&= \sum_{i=1}^{N}\left(6N - 6(i-1) + 4\right) + 2 \\
&= \sum_{i=1}^{N}\left(6N - 6i + 10\right) + 2 \\
&= 6N^2 - 6\frac{N(N+1)}{2} + 10N + 2 \\
&= 3N^2 + 7N + 2
\end{aligned}
$$

b) [6 points] To what big-Θ complexity class does your answer to the previous part belong? (No proof is necessary.)

Obviously, it is $\Theta(N^2)$.

2. [16 points] Consider the following two functions:

$$f(n) = n^2 \log n \text{ and } g(n) = n^3$$

Determine the complexity relationship between the two functions. That is, determine whether f is strictly $\Omega(g)$, or f is strictly $O(g)$, or f is $\Theta(g)$. Whichever conclusion you reach, state it clearly and justify your conclusion completely by applying relevant theorems from the course notes.

Since the first function is not listed in Theorem 5, we must apply Theorem 8 and its Corollary:

$$\begin{aligned}
\operatorname*{limit}_{n\to\infty} \frac{f(n)}{g(n)} &= \operatorname*{limit}_{n\to\infty} \frac{n^2 \log n}{n^3} \\
&= \operatorname*{limit}_{n\to\infty} \frac{\log n}{n} \\
&= \operatorname*{limit}_{n\to\infty} \frac{\frac{1}{n \ln 2}}{1} \\
&= 0
\end{aligned}$$

By the Corollary to Theorem 8, $n^2 \log n$ **is strictly** $O(n^3)$**.**

3. [16 points] When designing a container implementation in C++, why is it generally desirable to provide two versions of the search logic, as in the specified BST template interface:

```
T* const Find(const T& D);
const T* const Find(const T& D) const;
```

Explain carefully why both are needed; it would be useful to provide hypothetical examples of client code to support your explanation.

The second form is necessary so that the client can call `Find()` in a context in which the container object has been declared as a `const` object. For example, the client may have created a container object in one function and then passed it to another function by constant reference:

```
void Foo(const BST<int>& Tree, . . . ) {
   . . .
   . . . = Tree.Find(. . .);
   . . .
}
```

Since `Tree` is `const`, the function `Foo()` can only call member functions of `Tree` that are declared with the `const` qualifier, and the call shown above would not be allowed unless there were a const version of `Find()`.

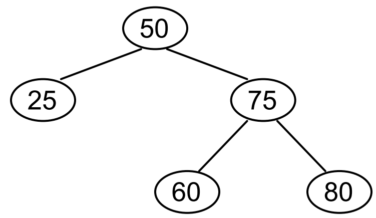
On the other hand, the call shown above would only be allowed if the pointer returned by `Find()` is assigned to a suitably `const` pointer:

```
   const int* const p = Tree.Find(. . .);
```

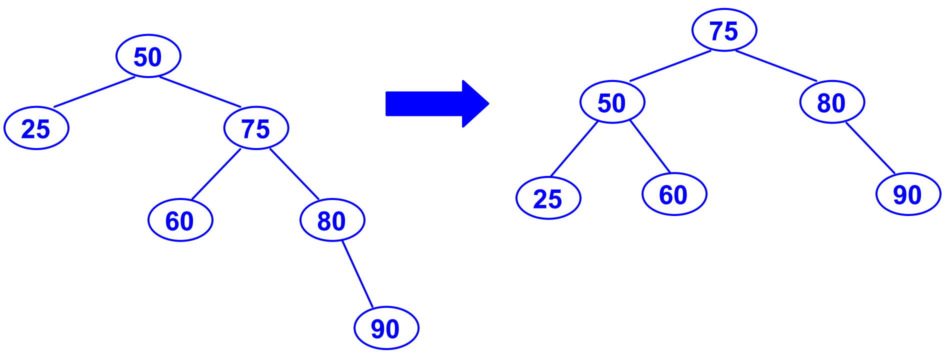
And, in this case, the client cannot use `p` to make any modifications to the target of `p` (which may actually be a good thing in most situations). But clearly in some cases, the client will want to call `Find()` to locate a data object and then modify that data object *in situ*. Therefore, we also need the first form of `Find()` to allows things like:

```
void DeleteEntry(unsigned int Offset, Location L) {
   . . .
   LocIdxEntry *p = QTree.Find(. . .);
   p->DeleteOffset( Offset );
   . . .
}
```

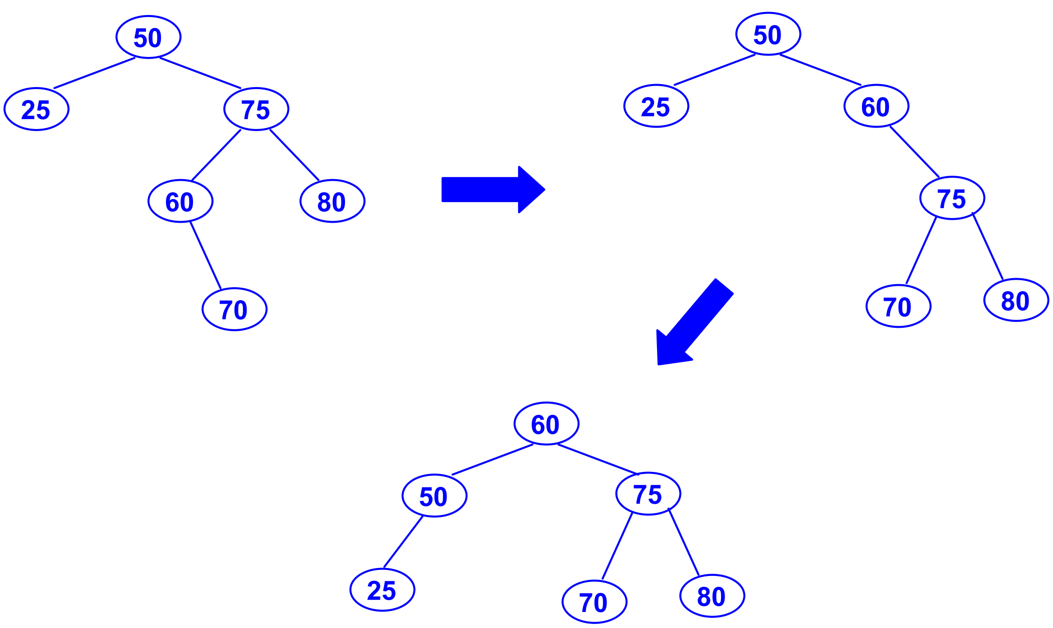
4. Consider the AVL tree at right:



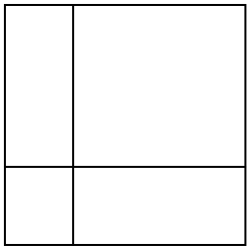
a) [8 points] Draw the resulting AVL tree if the value 90 is inserted.



b) [8 points] Draw the resulting AVL tree if the value 70 is inserted (into the original tree).



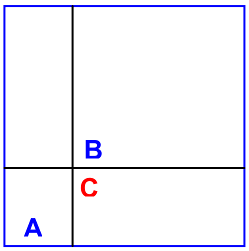
5. The PR-quadtree partitions a finite, square region into four identical sub-regions (quadrants). An alternate type of quadtree, let's call it an R-quadtree, allows unequal partitioning of sub-regions based on the actual locations of data objects, as shown below.

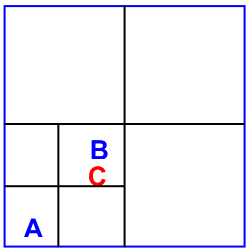


a) [8 points] Given the same set of data points, is it possible that insertion into the PR-quadtree would require splitting but insertion into the R-quadtree would not? If yes, give an example to illustrate how this could happen. (A clear diagram would be sufficient.) If not, explain why not.

It should be obvious that either type of tree will store a single data point without any splitting, and that both will require at least one split when a second data point is inserted.

However, after that, things may be different; suppose that A and B are inserted first:

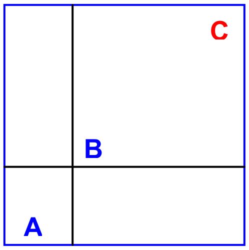


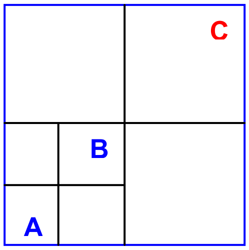


The PR-quadtree (on the right) required two splittings, while the R-quadtree (on the left) required only one. But that's not exactly what the question was asking (both DID require a split when B was inserted). However when C is inserted the PR-quadtree must split again, but the R-quadtree already separates the values.

b) [8 points] Given the same set of data points, is it possible that insertion into the PR-quadtree would require splitting but insertion into the R-quadtree would not? If yes, give an example to illustrate how this could happen. (A clear diagram would be sufficient.) If not, explain why not.

Take the same starting point as above, but place the third point C in a different spot:





Now, the PR-quadtree does not require performing another split, but the R-quadtree does.

6. [16 points] Recall the definition:

> Let f and g be non-negative functions of n. Then f is $\mathrm{O}(g)$ if and only if there exist constants $N > 0$ and $C > 0$ such that, for all $n > N$, $f(n) \le Cg(n)$.

Prove the following fact: if f, g and h are non-negative functions of n, and f is $\mathrm{O}(g)$ and g is $\mathrm{O}(h)$ then f is $\mathrm{O}(h)$.
Note: you may NOT use the theorem that states that big-O is transitive.

proof: Suppose that f, g and h are non-negative functions of n, and f is $\mathrm{O}(g)$ and g is $\mathrm{O}(h)$.

Then from the definition, there exist constants $N > 0$ and $C > 0$ such that, for all $n > N$, $f(n) \le Cg(n)$.
And, there also exist constants $M > 0$ and $D > 0$ such that, for all $n > M$, $f(n) \le Dg(n)$.

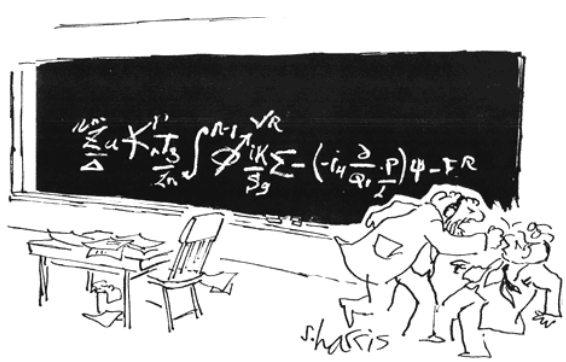
(Note: there is no reason to suppose that the same constants will apply for both relationships.)

Let $R = \max(N, M)$ and let $E = CD$. Then we have that, for all $n > R$:

$$
\begin{aligned}
f(n) &\le Cg(n) && \text{since } n > R \Rightarrow n > N \\
&\le CDh(n) && \text{since } n > R \Rightarrow n > M \\
&= Eh(n) && \text{since } E = CD
\end{aligned}
$$

Therefore, by definition, we have that f is $\mathrm{O}(h)$.

QED



"You want proof? I'll give you proof!"