READ THIS NOW!

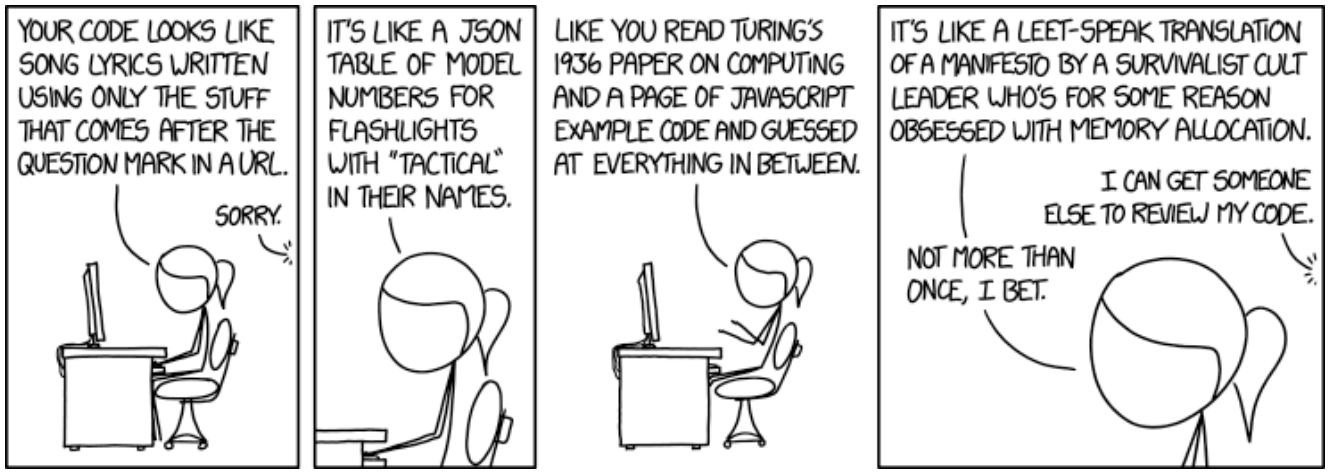
- Print your name in the space provided below.
- There are 3 short-answer questions, priced as marked. The maximum score is 50.
- Most questions require analyzing a scenario, drawing a conclusion, and justifying that conclusion. Credit will be awarded for making statements that are both true and relevant to the question. Do not feel obligated to fill all the available space when answering the questions.
- This examination is closed book and closed notes.
- No calculators, cell phones, or other computing devices may be used. The use of any such device will be interpreted as an indication that you are finished with the test and your test form will be collected immediately.
- Until solutions are posted, or gone over in class, 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.

Name (Last, First) Solution

printed

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

signed

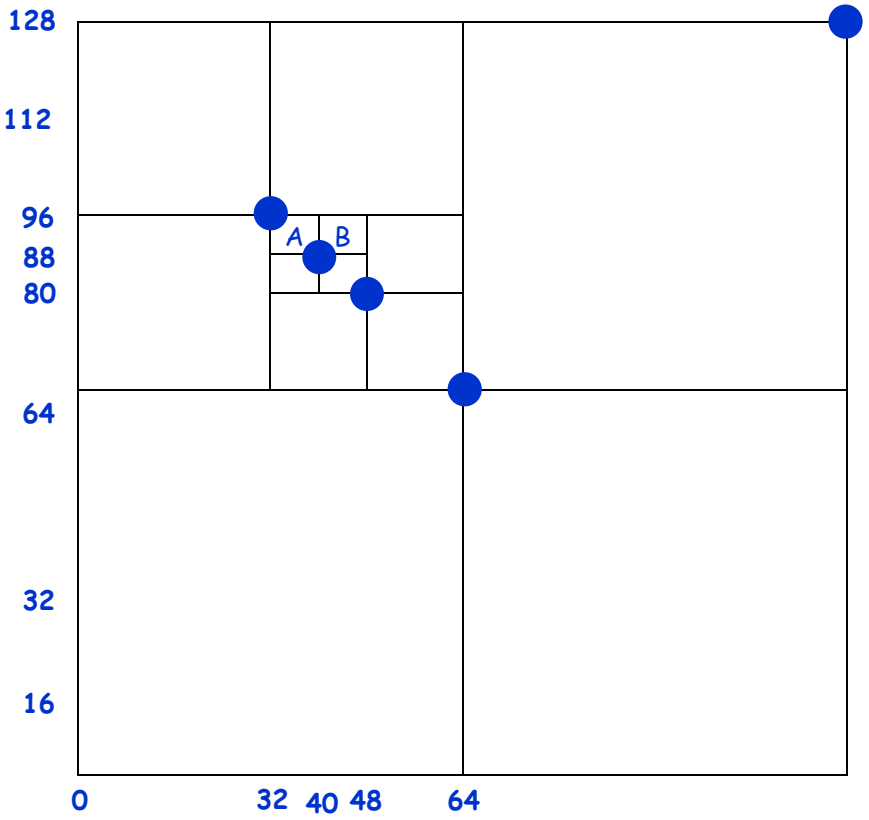

YOUR CODE LOOKS LIKE SONG LYRICS WRITTEN USING ONLY THE STUFF THAT COMES AFTER THE QUESTION MARK IN A URL.
SORRY.
IT'S LIKE A JSON TABLE OF MODEL NUMBERS FOR FLASHLIGHTS WITH "TACTICAL" IN THEIR NAMES.
LIKE YOU READ TURING'S 1936 PAPER ON COMPUTING AND A PAGE OF JAVASCRIPT EXAMPLE CODE AND GUESSED AT EVERYTHING IN BETWEEN.
IT'S LIKE A LEET-SPEAK TRANSLATION OF A MANIFESTO BY A SURVIVALIST CULT LEADER WHO'S FOR SOME REASON OBSESSED WITH MEMORY ALLOCATION.
I CAN GET SOMEONE ELSE TO REVIEW MY CODE.
NOT MORE THAN ONCE, I BET.

xkcd.com

1. Suppose we are building a PR quadtree, with world boundaries 0 <= x <= 256 and 0 <= y <= 256, and bucket size 1. For each of the following parts, partial credit will be assigned only if you provide an explanation or show clear, supporting work.

a) **[10 points]** We begin by inserting the points A(40, 90) and B(42, 90). How many splits will occur when B is inserted? Why? A diagram might be useful in explaining your answer.

The insertion of A results in a root that's a leaf node, and no splits. When B is inserted, we must split the world and A and B both fall in the SW quadrant of the world (so that's one split). After that:



Blue circles indicate centerpoints of regions that have been split.

The point A falls on a region boundary; if you assume it goes to the West, 5 splits are enough. If you assume it goes to the East, one more split is needed.

So… that's a total of 5 region splits in order to separate A and B (assuming that the right boundary of a region belongs to that region; if not, we need one more split).

b) **[10 points]** Next, we insert the following points:

C(100, 75) , D(100, 25) , E(25, 75) , F(200, 20)

How many leaf nodes will the PR quadtree contain? Why?

7

There are seven data values in the tree, and in a PR quadtree, every data object is in a leaf, and here each leaf can store only one data object.

2. [10 points] A student implements a Java class **`Foo`**. Here's part of the implementation:

```
public class Foo {

   private Integer ID;
   private String  Name;

   . . .

   public boolean equals(Object other) {

      if ( !this.getClass().equals(other.getClass()) ) return false;

      return ( this.ID.equals(other.ID) && this.Name.equals(other.Name) );
   }
}
```

Assume that the parts of the implementation that are not shown are correct (syntactically and logically). Also assume that the student's concept of what it means for two **`Foo`** objects to be equal is sensible, and reflected in the logic of the given implementation of the **`equals()`** method shown above.

Explain why the student's implementation of **`equals()`** would be unsatisfactory. Be precise and complete. Do not write corrected code! Explain!

There are several issues:

- there is no null test on the parameter, so a NullPointerException will occur if other == null
- other is a reference of type Object, so other.ID and other.Name will name nonexistent members and result in a compile-time error; other must be cast to obtain a reference of type Foo

Some non-issues:

- no check whether this == null; that's not even possible
- missing parentheses on first call to getClass(); treated as a don't care
- accessing private members ID and Name of Foo class; this is a member function of that class, so it can access all members directly
- comparing ID fields with equals() rather than ==; ID is of type Integer, so it's an object of a class; objects of a class are compared with equals(); primitives are compared with ==.

3. [20 points] There is a binary search tree, *T*, storing many integer values, none of which are duplicates. Among the values in *T* there are two integers *x* and *y*, such that $x > y$. *x* is stored in the node *Nx* and *y* is stored in the node *Ny*. We are not told which of these values was inserted first.

Give a precise, complete description of each of the possible physical relationships between *Nx* and *Ny*. One hypothetical relationship between two nodes would be " *Nx* is the parent of *Ny*" (not that I'm saying that's a correct answer). There are other, more complex relationships two nodes might have. In your answer, consider whether the order in which *x* and *y* were inserted would matter.

Suppose *x* was inserted first. When *y* is inserted, we will encounter a value that's larger than or equal to *y*, either before we reach *Nx*, or when we reach *Nx*. And, when that happens, the insertion of *y* will proceed to the left side of that node.

So, if *x* was inserted first, then *Ny* must either be in the left subtree of *Nx* or in the left subtree of some node along the path from the root to *Nx*.

Suppose *y* was inserted first. When *x* is inserted, we will encounter a value that's less than *x*, either before we reach *Ny*, or when we reach *Ny*. And, when that happens, the insertion of *x* will proceed to the right side of that node.

If *y* was inserted first, then *Ny* must be in the left subtree of some node along the path from the root to *Nx*.

Summing up, *Ny* must be in the left subtree of some node between the root and *Nx*, and if *x* was inserted first *Ny* might be in the left subtree of *Nx*.

The most common issue with answers was a lack of precision; for example, not being clear about left vs right.

The second most common issue was with answers that concluded that Nx and Ny must have a parent-child relationship. They do not.

Aside from that, few answers made clear that Nx and Ny might be separated by a "fork" at a node that was an ancestor of both.

And… "ancestor" nodes lie between a node and the root; "descendant" nodes lie in a subtree of a node.