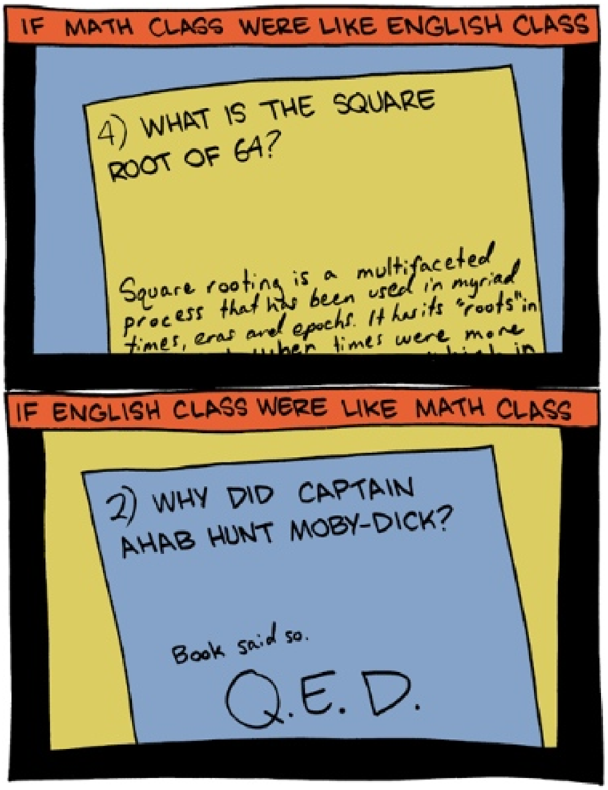
CS140 - Assignment 5

Due: Sunday, Oct. 9 at 11:59pm


IF MATH CLASS WERE LIKE ENGLISH CLASS
4) WHAT IS THE SQUARE ROOT OF 64?
Square rooting is a multifaceted process that has been used in myriad times, eras and epochs. It has its "roots" in
when times were more
IF ENGLISH CLASS WERE LIKE MATH CLASS
2) WHY DID CAPTAIN AHAB HUNT MOBY-DICK?
Book said so.
Q.E.D.

http://www.smbc-comics.com/index.php?db=comics&id=1872

You may (and are encouraged) to work with a partner on this assignment. **However, if you decide to work with a partner on this assignment, it must be someone that you haven't previously partnered with on an assignment.** If you're looking for a partner, post on slack (or email me). Please do this sooner than later!

0. *Optional:* We're about a third of the way through the course and I wanted to checkin and see how things are going. I've gotten some feedback the group assignments, but I'd also love to get individual feedback on things. If you want (it's anonymous), take 5 minutes and let me know how things are going:

   `https://forms.gle/yVdT3MEfRCUw7gWF8`

1. [**20 points**] More Balanced Binary Search Trees

   Red-black trees maintain a set of five properties; in class we showed that maintaining those properties guarantees that the height of a red-black tree with n nodes is never more than

$2\log(n+1)$. Consider another balanced binary search tree which maintains the following invariant: for any node x, the heights of the left and right subtrees of x differ by at most 1. We'll call these 1off trees.

(a) [14 points] Prove by strong induction on the height of the tree that a 1off tree with height h has at least $f(h)$ nodes, where $f(h)$ is the hth Fibonacci number. (Recall that $f(n) = f(n-1) + f(n-2)$ and that $f(0) = f(1) = 1$.)

(b) [3 points] Use the previous part to show that a 1off tree with height h has at least $2f(h-2)$ nodes.

(c) [3 points] Finally show that a 1off tree with n nodes has $O(\log n)$ height.

2. **[18 points]** Stacks and queues

Assume you're given an implementation of a stack that supports `push` and `pop` in $O(1)$ time. Now you'd like to implement a queue using these stacks.

(a) [10 points] Explain how you can efficiently implement a queue using two of these stacks. ("Efficiently" means in a way that allows you to do the next part of the problem.)

(b) [8 points] Prove that the amortized cost of each `enqueue` and `dequeue` operation is $O(1)$ for your stack-based queue by using the the aggregate amortized analysis technique.