CS161 Midterm Test Corrections

Please read the entire document thoroughly before beginning your test corrections.

What you can submit test corrections for

You may submit test corrections for any problems in Sections 3, 4, 5, 6, or 7. We will not be accepting test corrections for Section 2, the multiple choice questions. You are not required to submit test corrections for every problem, so feel free to submit test corrections for as many or as few problems as you’d like. PLEASE DO NOT SUBMIT TEST CORRECTIONS FOR PROBLEMS WHERE YOU BELIEVE THERE’S BEEN A MISTAKE IN GRADING – SUBMIT A REGRADE REQUEST ON THE ORIGINAL MIDTERM INSTEAD.

What you should submit

You will need to submit a typed PDF containing your test corrections. We will be strict on the formatting requirements for your test corrections, so please read this section carefully. For any one of problems 3.3, 3.4, 3.5, 3.6, 3.7, 4.1, 5.2, 6.1, 6.2, and 7, please provide and clearly label the following four components:

1. **Number of points you originally earned and the rubric item(s).** For example, please write: I earned 2 points (out of 5) on 3.6. Rubric item: “Right answer but significant error in explanation, OR wrong answer but made an incorrect assumption (where the wrong assumption made was clear or stated)”. If you received additional point adjustments or rubric items, please list them as well.

2. **Revised answer.** Please submit a new, complete answer, even if it overlaps with your original answer.

3. **Mini reflection.** If your deduction was –1 point (or less), then you may skip this part. Otherwise, please share with us in ~5 sentences or less (or more if you had a lot to reflect on!):
   - **What the mistake was.** Think critically about this (please don’t just say “the mistake was that the submitted solution was wrong”). For example, did the submission assume that a particular algorithm could be applied to the problem and achieve the desired runtime when the problem constraints might not guarantee that desired runtime? Did the submission fail to account for a particular case or aspect of the problem? Was it too vague about an important aspect of the problem? Or was it something larger, such as not recognizing that the problem was quite similar to a section/homework problem, and that similar strategies may be helpful?
   - **Any advice you’d like to share with your pre-midterm self or with a future CS161 student who’s studying for the midterm.** We want to make sure that students feel like they’re getting the full value/purpose from the problem, and we hope that students feel that revisiting their midterm answers is productive. Whether it’s remembering which aspects of the question should have served as clear signals for certain algorithm design paradigms, or tips for what to double check when assessing the completeness of a proof or solution, please share with us any and all takeaways you think would be helpful to other students (if it helps, imagine you’re a CA trying to emphasize learning goals to a student).

4. **Original answer.** Simply include an exact copy of your original answer (a screenshot or image is fine). As with the other 3 components, please label this section especially clearly, so that a grader doesn’t accidentally mistake it for your new revised answer.
Collaboration & asking questions

Test corrections must be completed individually. You are not allowed to discuss any midterm questions or solution strategies with your peers. Please treat this test correction period as an extension of the take-home exam period, so the same collaboration and honor code policies apply. Similarly, CAs cannot help with identifying your errors or provide further hints on these problems, but we are still happy to answer clarifying or logistical questions via private Piazza posts.

How we’ll grade your test corrections

When grading your test corrections for a particular problem, we’ll either award you half your deducted points back, or return 0 points to you. In other words, you should expect no partial credit grading for test corrections. To earn half of your deducted points back on any problem, you must:

- Submit a revised solution that would have earned full credit.
- Include all the necessary components mentioned in the above section. In particular, your reflection needs to demonstrate that you made a thoughtful and concerted effort to identify lessons learned and areas of improvement.

If your revised solution still contains errors, if you do not include one of the components above, or if your reflection doesn’t show much effort, then we will not return any points back to you for that problem.

How test corrections will be factored into your final grade

First, I will generate final letter grades for everyone as if test corrections didn’t exist (i.e. using original midterm scores), as discussed on the website. Second, I will generate a second set of final letter grades as discussed on the website, except everyone’s midterm scores will include any points returned via test corrections. Each student will receive the maximum of these two letter grades. Note: this is a Pareto-improving change—no one will receive a worse letter grade than they would if test corrections were not a component of the course. Basically, you should really feel reassured that not doing test corrections (or only choosing to do test corrections for a couple problems) cannot possibly hurt you.

Midterm solutions & regrade requests

We will release midterm solutions once all test corrections are received. Now that midterm grades are released, we will be accepting regrade requests on the original midterm submission until next Wednesday (7/29) 11:59pm. As mentioned already, if you believe there was a mistake in grading, submit a regrade request instead of doing test corrections.

Some tips & advice

We recommend not worrying too much about getting test corrections in for every single problem—since we expect very thoughtful corrections, focusing on quality over quantity will serve you well here, especially if you don’t have a lot of time.
Here are some tips and hints to help you get you started on a number of the problems. Each of these is related to common errors we saw; if a problem isn’t listed, this isn’t because it is easy, but because the tips we’d give are those found in the problem statement, or because we found that different students submitted solutions with very different errors. **We will not be providing further hints about any questions on Piazza or in Office Hours.**

- **3.5.** Read the problem carefully! Note that i and j are not constants.
- **3.6.** Remember, you’re allowed to be creative with how your algorithm processes the unsorted array to produce a BST.
- **3.7.** Read the problem very carefully. We are looking for a worst-case runtime of \(O(n)\). k is not a constant.
- **5.1.** Many people missed a crucial component of the induction proof. A lot of you have the inductive hypothesis and set-up down, but we were specifically looking for solutions that explicitly address *why* the child recursive call’s answer can safely be returned by the parent recursive call.
- **6.2.** Since we explicitly do not guarantee that the maximum year value is any particular function of \(n\) (it’s unbounded, so it could be \(n^i\), for example), please be careful about the kinds of in-class algorithms you choose to deploy as part of your solution. To get you on the right track, think (deterministic) divide-and-conquer. Remember, this is a 12 point question that was meant to be significantly more involved than 6.1, which is more of a warmup.
- **7.** The hint says it all. Be very careful about what your recursion tree looks like. Also, pay attention to the space constraint, and the note about hashing. You should also be spelling out very clearly (and in pseudocode) how you ever check if elements appear \(k\) or more times. The solution we have in mind lends itself to pretty straightforward pseudocode, and it’ll be easy for you to ensure that you only ever maintain a constant number of counts. And remember, k is *not* a constant!