General tips

TABLE OF CONTENTS

1 Planning
2 Implementation
3 Getting help

Planning

- **Start early.** This homework requires a lot more design than previous homework assignments and becomes much easier when you allocate some time for thinking carefully about how you want to implement it.
- **Read the entire spec first.** Unlike previous homework assignments, this homework has subparts that depend closely on one another. It will be difficult to bolt on new features to an implementation that doesn't take into account the entire behavior of the system.
- **Do not jump right into coding.** It is easy to lose track of the bigger picture when you jump into the code immediately. Writing even an extremely short design document will go a long way in avoiding difficult-to-find bugs down the line.
- **Design for simplicity.** We are not testing you on your ability to make the most efficient or elegant solution. When deciding which design to implement, choose the one that you know you can implement correctly and you are sure satisfies the necessary constraints even if it does not seem to be optimal. A design you can understand will be easier to debug than a more elegant design that you do not fully understand.

Implementation

- **Thoroughly test each part.** Make sure that each section you implement works exactly as you intend it to before moving on to the next part by inspecting all of the relevant state with logging statements. This will make it significantly easier to narrow down the location of a bug that shows up later on as you can assume with some certainty that it is not in the parts you tested individually.

- **Use print statements first, then debug your logic.** In this homework, the code you write will be relatively short and is distributed among several RPCs that can be inspected individually. We recommend first using print statements to determine which RPC seems to be behaving incorrectly, then going through your implementation of that RPC line by line and making sure that the behavior of the code matches the behavior you intend. Going through the spec carefully and comparing it to your code's intended behavior can also help catch several bugs.

Getting help

- **Run your ideas by TAs and other students.** While you may not share code or implementation-level details with other students, it is totally fine to share high-level ideas on how to implement the desired behavior. This will allow you to make sure there are no flaws in your understanding of the assignment and see if there are better ways of going about the solution.
- **Ask what worked and didn't work for others.** If you are stuck on a hard-to-find bug and reviewing the Detailed testing breakdown section did not help, you may ask other students or TAs what errors they had that caused similar bugs on Ed or in OH. Keep in mind that you may not have other students look at your code for debugging purposes, but there are several common logical mistakes that other students might have encountered as well.
