About how long did Exercise 12 take you?

A. [0, 2) hours
B. [2, 4) hours
C. [4, 6) hours
D. [6, 8) hours
E. 8+ Hours
F. I didn’t submit / I prefer not to say
Thinking About Systems
CSE 333 Summer 2023

Instructor: Timmy Yang

Teaching Assistants:
Jennifer Xu  Leanna Nguyen  Pedro Amarante
Sara Deutscher  Tanmay Shah

Partially adapted from James Wilcox’s Wi23 CSE452 lecture, “Performance and Queuing Theory”
Relevant Course Information

❖ Homework 4 due Wednesday (8/16)
  ▪ Submissions accepted until Friday (8/18)

❖ Course evaluations (see Ed #404) due Friday night
  ▪ Please fill them out. They help all staff members improve their skills as educators and allow us to improve the course for future offerings. 😊

❖ Quiz 4 open Wednesday (8/16), closes Friday (8/18)

❖ Next lecture (8/16) will be extra OH/work day.

❖ Check grades as Canvas assignments are released
  ▪ Assignments with regrades closed will get transferred first.
Today’s Goals

❖ We’ll be trying to put the “System” in “Systems Programming”

❖ The systems motto is: Is the problem the system solves real?
  ▪ Does the system solve the problem?
    • Is the system correct?
    • Is the system fast enough to be useful?

❖ Correctness is the most important part of system design, but performance is important too.

❖ Won’t be explicitly covered on Quiz 4
  ▪ More of a bonus lecture 😊
Aside: Talking about Performance

- Performance is a bit of a loaded term.
- What do we mean when we say something “performs well?”
  - Does it go fast?
    - Throughput? Latency?
  - Is it memory efficient?

- Ignores other important metrics:
  - Is the system accessible?
    - Is it easy for everyone to use?
  - Is it easy to understand?
  - If things go wrong, can a user fix it themselves (i.e., can you tolerate error)?
  - Is this system cost-effective (i.e., can we afford it)?
    - From space, size and monetary stand-points.

- When designing systems, it’s important to consider more than just, “going fast.”
So... What is a System?

❖ “A **system** is a group of interacting or interrelated entities that form a unified whole. A system is delineated by its spatial and temporal boundaries, surrounded and influenced by its environment, **described by its structure and purpose and expressed in its functioning.**”

- Still vague, maybe still confusing

❖ Put a little more simply, “...a set of interconnected components whose joint behavior is observed at the interface with its environment.”
Systems

What kind of systems can you think of (computing or otherwise)?
Systems from 10,000 Feet
Systems from 10,000 Feet

❖ The interface is a box separating the components of the system from “everything else”
  ▪ Outside of the box is “the environment”
  ▪ Inside the box is the system, including its components and connections

❖ We observe the system at its boundary with the environment
  ▪ Observers don’t look inside the box; they look at the behavior of the box.

❖ Notice that this definition does not contain the word “computer” or even “electrical signals”
  ▪ Applies equally well to biological, mechanical and social systems
Systems from 10,000 Feet

❖ You largely interact and view systems from an observer perspective.
  ▪ As a user, client, etc.
  ▪ i.e., “outside looking in”

❖ “Systems Programming” is (ideally) all about starting to peel away the interface barrier.
  ▪ We want to understand the inner workings of systems and how to build/design them.
  ▪ All about understanding “the layer below.”
Analyzing Systems

❖ What have we thought about while developing systems?
  ▪ Consider both the perspective of an observer and a developer

❖ On the Observer Side:
  ▪ Inputs and Outputs
    • What do we give the system, what does the system give us?
  ▪ Interface
    • How do we interact with the system?
  ▪ Robustness and Error Messaging
    • What errors or unexpected behavior can we tolerate and how?
    • If something goes wrong, how do we let people know?
Analyzing Systems

❖ What have we thought about while developing systems?
   ▪ Consider both the perspective of an observer and a developer

❖ On the Maintainer/Developer side:
   ▪ Everything from Observer side plus...
   ▪ Interface (again)
     • What am I allowing the user to do or not to do?
   ▪ Correctness and Performance*
     • Does the system do what we want it within time and resource constraints?
   ▪ Maintainability
     • How easy is the system to fix and update?
   ▪ Design and Changes
     • Who gets a say when designing and/or changing the system?

*Won’t be focusing too much on this point, as it already gets plenty of discussion in the CS space.
The Computer as a System

- Modern computer systems are increasingly complex!
  - Networking, multiple CPU cores, various peripherals, etc.
  - Multitudes of operating systems, applications, etc.
The Computer as a System

❖ Let’s analyze a computer as an observer!
  ▪ What are our inputs?
    • Electricity, keystrokes, data from the network, etc.
  ▪ What are our outputs?
    • Text on a terminal, video feed out, audio from speakers.
  ▪ How do we interact with the system?
    • Keyboard, mouse, peripherals that can connect through ports.
  ▪ What errors can we tolerate/handle?
    • Some memory corruption, bad components, etc.
  ▪ If something goes wrong, how do we know?
    • “Blue Screen of Death”
The Computer as a System

❖ Now what about as a maintainer/developer?
  ▪ Interface
    • Physical hardware, software tools, programming languages
  ▪ Maintainability
    • Documentation, hardware specifications, software updates
  ▪ Design and Changes
    • Hardware standards, various protocols
      – Aside: RFCs, and RFC 1149 – IP over Avian Carriers
    • Who has a say in all of this?
      – Tech companies
      – Hardware manufacturers
Allen School as a System

❖ Your turn! Let’s think about the Allen School as a system.
  ▪ Consider the system as both an observer and a maintainer.
  ▪ Open-ended activity, no correct answers!

❖ As an observer:
  ▪ Inputs and outputs
  ▪ Interfaces
  ▪ Robustness and Error Messaging

❖ As a maintainer:
  ▪ Interfaces
  ▪ Maintainability
  ▪ Design and Changes
Allen School as a System

❖ As an observer:
   - Inputs and Outputs:

   - Interfaces:

   - Robustness and Error Messaging
Allen School as a System

❖ As a maintainer:
  ▪ Interfaces:
    ▪ Maintainability:
    ▪ Design and Changes:
Interfaces

❖ We’ve talked a lot about interfaces this quarter.
  ▪ What information can we give/get through the interface?

❖ Interface design is an important part of designing a system.
  ▪ UX Designers exist for a reason!

❖ Interface and system design can also drive user behavior in interesting and surprising ways
  ▪ HW1 vs STL: LinkedList vs std::list
  ▪ HW2/HW3 vs HW4: Command-line vs web-based
  ▪ Editors: VSCode vs. vi(m) vs. emacs
Real-World Systems and Interfaces

❖ Most of these ideas can be applied to many other real-world (non-computing related) systems.

❖ We’ll be focusing on the US Electoral System
  ▪ Specifically on voting.

❖ Note: Political topic by nature of subject matter. My own views do not reflect those of the Allen School.
Voting

❖ Elections are a complex topic, let’s focus on how most people interact with this system: voting

❖ Voting generally conducted in-person, some states allow for sending in votes by mail.

❖ Mail-in Voting
  - 27 states and Washington D.C. offer “no-excuse” absentee voting
    - 8 states conduct elections entirely by mail (e.g., Washington)
  - Leaves 15 states that require you to be out-of-state, sick, etc. to vote by mail.
Voting

- 15 states where you must show up in-person to vote at a voting machine unless you have an acceptable excuse.

Accessibility:
- Must be able (and have the time) to show up to a voting location.
- What happens if you don’t feel safe at said voting location?
  - “GOP activist group instructs Michigan poll watchers to call 911”

Design and Changes:
- Once you’re in a position of power, you can influence the drawing of electoral districts.
  - At best, things are fairer and represent the population well.
  - At worst...
Gerrymandering

- Manipulation of electoral district boundaries with the intent to create an advantage for a party, group or socioeconomic class within the constituency
  - CGP Grey on Gerrymandering: [https://youtu.be/Mky11UJb9AY](https://youtu.be/Mky11UJb9AY)

- Manipulate how voting districts are decided to put others at a disadvantage, or to put yourself at an advantage.
Redlining

“**Redlining** is a discriminatory practice in which services (financial and otherwise) are withheld from potential customers who reside in neighborhoods classified as "hazardous" to investment; these neighborhoods have significant numbers of racial and ethnic minorities, and low-income residents.”


- Push certain demographics into certain zones. Make it harder for them to access resources.
  - Easier time staying in office!
Electoral System Analysis

❖ With all of this in mind, let’s analyze this a bit closer.

❖ First, as an observer:
  ▪ Inputs and outputs
    • Votes, policies, elected officials
  ▪ Interface
    • Ballots, voting machines
  ▪ Robustness and Error Messaging
    • Must explicitly register to vote (citizenship, etc. checked then)
    • Recounts and audits when suspicion of fraud, miscounts, etc.
Electoral System Analysis

❖ With all of this in mind, let’s analyze this a bit closer.

❖ As a maintainer:
  ▪ Interfaces
    • Constituents can vote on subject matter, and government officials
    • Can also directly contact officials through various channels
  ▪ Maintainability
    • *Very* hard to change, requires many people to agree, and then enough votes.
  ▪ Design and Changes
    • Heavy influence by those that are already in power
    • Influence from those that aren’t in government through votes...
Wrap-up

- Systems are complex, and there’s a lot to think about.
  - Does it solve the problem?
    - Consistently and in a reasonable time?
  - Is it accessible and user-friendly?
    - Will people want to keep using it? Will people be *able* to use it?
  - Is it easy to understand and maintain?
    - Will people want to keep *updating* it?
  - Who gets a say in its design? Now? In the future?

- As programmers there is a high likelihood that you’ll be designing some of your own systems in the future.
  - Keep some of these concepts in mind while designing systems 😊