15-213: Introduction to Computer Systems
Written Assignment #8

This written homework covers Networking and Concurrency

Directions

Complete the question(s) on the following pages with single paragraph answers. These questions are not meant to be particularly long! Once you are done, submit this assignment on Canvas.

Below is an example question and answer.

Q: Please describe benefits of two's-complement signed integers versus other approaches.

A: Other representations of signed integers (ones-complement and sign-and-magnitude) have two representations of zero (+0 and −0), which makes testing for a zero result more difficult. Also, addition and subtraction of two's complement signed numbers are done exactly the same as addition and subtraction of unsigned numbers (with wraparound on overflow), which means a CPU can use the same hardware and machine instructions for both.

Grading

Assignments are graded via *peer review:*

1. Three other students will each provide short, constructive feedback on your assignment, and a score on a scale of 1 to 20. You will receive the maximum of the three scores.
1. You, in turn, will provide feedback and a score for three other students (not the same ones as in part 1). We will provide a *rubric*, a document describing what good answers to each question look like, to assist you. You receive five additional points for completing all of your peer reviews.

Due Date

This assignment is due on Wednesday, August 3, 2022 by 11:59pm Pittsburgh time (currently UTC−4). Remember to convert this time to the timezone you currently reside in.

Peer reviews will be assigned roughly 12 hours later, and are due a week after that.

Question 1 (3 points)

Explain the stacked (layered) architecture of the internet protocols. List an advantage and a disadvantage of this architecture.

Explanation:
The stacked architecture of internet protocol consists of many protocols that interact with the protocols just above and below the layer of the given protocol. Each layer of a specific network model may be responsible for a different function of the network. Each layer will pass information up and down to the next subsequent layer as data is processed.

Advantages:
->Interoperability and allows for so many protocols supported by the current internet.
->Portability - Layered networking protocols are much easier to port from one system or architecture to another.
->Compartmentalization of Functionality - The compartmentalization or layering of processes, procedures and communications functions gives developers the freedom to concentrate on a specific layer or specific functions within that layer's realm of responsibility without the need for great concern or modification of any other layer.
(Any answer that closely aligns to one of the advantages mentioned above can be granted marks)

Disadvantage - Overhead increased due to headers of each layer

Question 2 (3 points)

Linux provides the functions *getaddrinfo* and *getnameinfo* to convert between `struct sockaddr` objects and string representations of addresses. Since every network-layer protocol has its own `struct sockaddr_PROTO`, different from all the others, how does *getaddrinfo* arrange things so its caller doesn't need to know which structs to use?

getaddrinfo returns a linked list of `struct addrinfo` objects. Each of these objects encapsulates a `struct sockaddr_PROTO` of the appropriate type. All of the information needed to call *socket* and *connect* (for a client) or *bind* (for a server) can be read directly out of a `struct addrinfo`, without knowing what type of socket address it wraps. When the caller of *getaddrinfo* is done with the list, it can use *freeaddrinfo* to deallocate it all at once.

Question 3: **Get RICH! :) or poor? :((4 points)**

The year is 1969. ATMs have just been invented. Thanks to ATMs, people don't have to go to banks anymore to deposit and withdraw money. But wait! Some people start complaining about their bank account not showing the correct balance. The bank hires a professional debugger. That's where you come in with your hat, pipe, magnifying glasses and a teddy bear (for debugging purposes only).

The bank tells you that the issue has mostly been seen in situations where a person was trying to withdraw money from the ATM and someone else was paying this person by depositing money to the same account from another ATM.

The ATM has 2 functions: Deposit and Withdraw.
Deposit has 5 steps:
D1 Count money to be deposited.
D2 Read the amount currently in the bank account.
D3 Add current balance and money to be deposited.
D4 Set amount to the amount calculated in step D3.
D5 Display success message.

Withdraw has 6 steps:
W1 Get amount to be withdrawn.
W2 Read amount currently in the bank account.
W3 If amount to be withdrawn > current balance. Exit with a failure message.
W4 Subtract amount to be withdrawn from the current amount.
W5 Set amount to the amount calculated in step W4.
W6 Display success message.

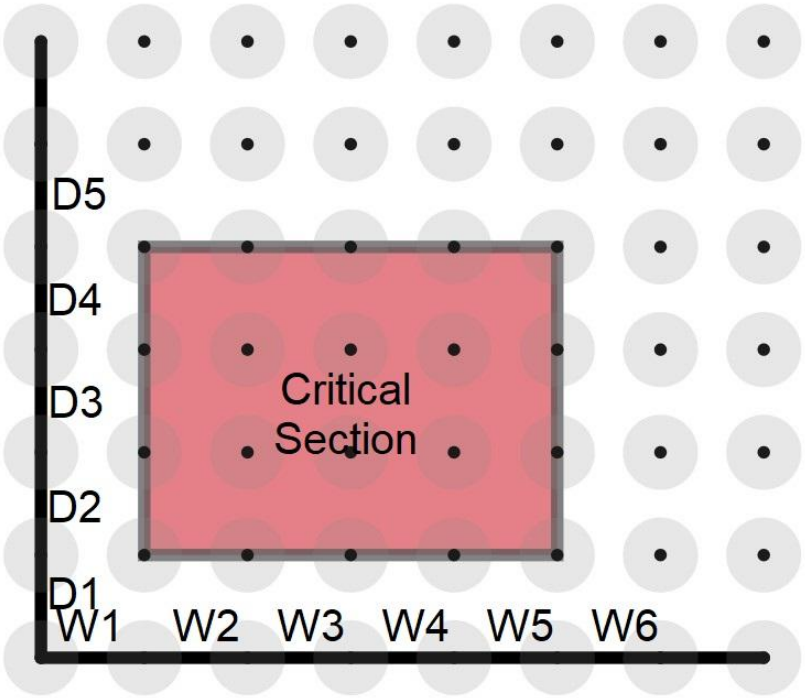
Answer the following questions:

A. What could be causing the issue?
B. Give a sequence of operations that would make someone richer than they should be.
C. Draw a progress graph of events when one person is withdrawing money from an account, and another depositing money to the same account, simultaneously. Indicate the critical region that must not be entered.
D. What synchronization primitive should be used to fix the bug?

A. It seems like the deposit and withdrawal steps are interleaving, leaving the system and the balance in the bank account in an inconsistent state.

B. Any sequence in which D2 happens between W1 and W5 and D4 happens after W5.

C. Steps D2 to D4 should be atomic and W2 to W5 should be atomic.
The following progress graph depicts the critical region which must not be entered.



D. A mutex is the most obvious choice, but any synchronization primitive that can implement mutual exclusion could also be used (e.g. a binary semaphore, or a rwlock where both D and W are writers).