What is Logic for Systems?

What programs have you written and enjoyed/found interesting? Did they work? How do you know if your program works? Does there exist a signal that tells you if your program works?

- Testing
- Grades
- TA Authority

Without TAs, who is the final authority on whether your code works?

- The user

Testing is not enough, it is a baseline. How do you reach a high level of assurance that your code works? Why doesn't testing give you that assurance?

- Testing doesn't involve the underlying logic
    - There could be edge cases that you may miss in testing
    - We can't debug the bugs we don't know about
- Program may not be deterministic

Program that accepts list of ints vs program that accepts int32s.

- There are 2^32 int32s
- There are infinite lists of ints
    - You can't test this exhaustively with a finite test suite

> Tests are a good foundation, but don't get us to that point of knowing vs believing

Puzzle: Alice, Bob, and Charlie work at the same company
Alice supervises Bob, who supervises Charlie
Employees of this company have graduated either from Brown or from a lesser school
Alice went to Brown, Charlie went to Harvale.
Based on this information, does someone who graduated Brown supervise someone who graduated from another school (assuming everyone graduates only from one school)?

- If Bob is a Brown grade, the answer is yes. If Bob did not go to Brown, then the answer is still yes since Alice supervises Bob.

Public key cryptography
Public and private key, you share public key with others but keep private key to yourself
> It is very difficult to decrypt a message encrypted with someone's public key, without also having their private key

Tim wants to negotiate shared secret with his bank.
Tim ---------- {Tim, S_Tim}_Bank----------> Bank
Tim <---------- {S_Tim, S_Bank}_Tim --------> Bank
Tim ----------> {S_Bank}_Bank ----------> Bank

At the end of this handshake, both sides have both halves of the secret, and no one else does.

It turns out that this protocol is broken:
There is an implicit assumption that there are only 2 people involved in the execution of the protocol (malicious third party, Eve).

Tim ----- {Tim, S_T}_E ----------> Eve ------- {Tim, S_T}_B -------> Bank
Tim ---------------------------------- Eve <------ {S_T, S_B}_T -------- Bank
Tim <----- {S_T, S_B}_T ---------- Eve ---------------------------------- Bank
Tim -------- {S_B}_E --------------> Eve ---------------------------------- Bank
Tim ---------------------------------- Eve ------ {S_B}_B --------------> Bank