Name: _____ Netid: _____

2022 Prelim. 5 questions
Each question is worth 20pts, divided equally among any sub-questions

Open book / notes, but you must not ask anyone else for help.
Released at noon, 3-8-22. Due by noon 3-11-22.

This should take 75-90 minutes, but we are giving you 3 hours to eliminate any possible time pressure. Just edit this word file, save as a PDF, and then upload the PDF version. We use logs to verify that you uploaded it within 180 minutes after downloading.

Type your answers and keep them short and clear – we aren't looking for essays.

Q1 problem setting: One very famous use of a key-value store (DHT) is to maintain the Amazon shopping cart. The system is called Dynamo. When you click to add something to your cart, a DHT put occurs, and the purchase is saved into a shard determined by your customer id (for example, "Purchase 1" by "Ken Birman" is saved in a shard obtained by hashing "Ken Birman"). Each different purchase has a separate id, so items won't overwrite one-another unless you revise a selection, such as by changing the size or the number of units: Amazon allows you to edit a purchase or delete it, before checking out. Checking out scans your shopping cart and then charges you for everything.

In their paper about this system, Amazon describes how a system failure sometimes makes it hard to reach the members of the correct shard. For this, Amazon does something unusual: they just increment the shard number until they find a shard where they *can* store the key-value item. Eventually, when the DHT recovers from whatever the issue might have been, items will automatically migrate back to the shard where they really should reside. For example, if "Ken Birman" normally maps to shard 6, maybe "Purchase 1" temporarily ended up on shard 7. But then the nodes in shard 6 recover, they notify the nodes in shard 7, and the key-value pairs get shifted from shard 7 to shard 6. There is no locking used when this kind of transfer occurs.

Amazon is a big believer in the CAP principle and in the BASE methodology.

a) State the CAP principle in your own words, ***briefly***. We already know that C stands for consistency, A for availability and P for network-partition tolerance. We want you to explain what those words mean, and what CAP is saying we should "do" when designing scalable, rapidly responsive systems.
b) BASE is short for basically available, soft state with eventual consistency. "Basically" means "to the degree possible." Explain what the words "available", "soft state" and "eventual consistency" mean, and then with one additional sentence, what BASE is recommending.
c) In words, tell us what Jim Gray's study tells us about transactions on a cloud-hosted database. Don't show us the formula – tell us what Jim's conclusions were and what they are based on.
d) Now, show us the formula Jim derived, and define the terms it uses. Tell us what it "means".
e) Suppose that we have a transaction that touches objects that live in different shards. For example, it reads A which is in shard X, then uses what it read to update B, in shard Y. What are we supposed to do when we follow Jim's advice? Will this impact the normal transaction guarantees? How?

Q2: Paxos Protocol. 5 parts.

Initially, for parts (a), (b) and (c), focus on Paxos with a fixed set of servers: N=3, Q_W=2 and Q_R=2.

a) Paxos was created as a solution to the state machine replication model. Define the model. Why can't we run Paxos with Q_W = N?
b) Why can't we just use Q_R=1?
c) Are there other possible values of Q_W and Q_R for this case of fixed N, with N=3? Explain.

Now for (d) and (e) suppose that Paxos has been deployed in a virtual synchrony membership context:

d) Virtual synchrony reports a new view if a process crashes or freezes up. Why does this enable Paxos to run with Q_W = N? Explain.
e) Virtual synchrony introduces the idea of state transfer. What state would we transfer to a joining process if we combine virtual synchrony with Paxos? Explain.

Q3: Some true/false questions. Put "T" or "F" in the box on the left to show your answer.

T/F	Assertion
	a) A stateless program needs to be coded in a functional language, like O'Caml because any variables to which values can be assigned while the code is running are a form of state.
	b) When we use state machine programming in the cloud, we have to implement logic to store and update the state, a mechanism to trigger state transitions, and the action that each state transition will initiate.
	c) Serialization of an object containing child objects requires design decisions such as whether each we should have one key-value pair per child object, versus recursively serializing child objects as part of the serialization of the parent object.
	d) If a sender machine and a destination machine use different CPU types (like Intel and AMD), data sent in messages will need to use a standard representation for objects like ints or doubles, resulting in overhead to translate in and out of the message format.
	e) The Byzantine failure model is used in Paxos libraries and systems like Derecho/Cascade
	f) To ensure availability even with 1 failure we must run Paxos on at least 5 servers
	g) A consistent cut is a set of instants in the timelines of the processes in a system, which could have occurred simultaneously in real-time.
	h) RDMA runs Paxos in the network routers.
	i) With Lamport's logical clocks, we can conclude $A \rightarrow B$ iff $LT(A) < LT(B)$
	j) With vector clocks, we can conclude $A \rightarrow B$ iff $VT(A) < VT(B)$

Q4: Consistent snapshots. Suppose that a banking application supports deposit, withdraw and transfer operations. Just like in homework 2, the bank is physically implemented by a sharded DHT for scalability, and different accounts could be on two different shards.

A transfer is done this way: using reliable message passing, the transaction is "sent" to the shard holding the source account, and it starts there by verifying that there is enough money in the account, then deducting the transfer amount. Next, again using reliable messaging, the transaction is forwarded to the destination account. Once it arrives, it is executed and will add the transfer amount to this account.

a) Lamport defined the "happens before" relationship, $\rightarrow$, in such a way that if event a might have caused event b, a $\rightarrow$ b. In our bank transfer operation, would it be correct to say that if w is the operation that withdrew money and d is the operation that deposited the transfer, w $\rightarrow$ d? Explain.
b) Suppose that b is included in a consistent snapshot, and a $\rightarrow$ b. Would we expect a to be included too, or might a be omitted from the snapshot? Explain, giving an example from the bank to help us visualize exactly what this issue is about.
c) We run a consistent snapshot algorithm. After it terminates, we have a set of files that represent the output – they constitute the actual snapshot. How do the numbers of files relate to the number of bank servers and network connections amongst them? What will the files contain?
d) Given a snapshot, we want to compute the total amount of money the bank is managing. Describe how we would calculate this total amount from the data in the snapshot.

Q5: Stateless first tier. In the cloud we use a *stateless* layer to handle client requests.

a) If we have N clients currently making requests, how many stateless functions will be running?
b) Define *elasticity* and explain why the first-tier is considered to be an elastic cloud layer.
c) Suppose that we want to keep a count of how many requests of a particular kind each customer makes, in order to charge for them (for example, in a game we might count purchases of extra strength points, or special weapons). Since the first tier is stateless, where would this state be kept?
d) Do we need locking to implement counters of the kind arising in (c)? Justify your answer.
e) Explain what a *container* is, and why container virtualization is popular for first-tier servers.