Name:Hanliang Jiang
NetID:hj33

Problem 1:

a. A blockchain is a decentralized digital ledger and it uses proof of work for transactions, Whereas cryptocurrencies are digital currencies that run on top of a blockchain.
b. Mining is the process of validating and adding transactions to the blockchain by solving complex mathematical problems; proof of work is a consensus algorithm to verify transactions by requiring users to demonstrate their work on complex mathematical problems.
c. Permissioned blockchains require authorization to join in, while permissionless blockchains do not need users to be authorized to join.
d. No. Bitcoin uses Proof of Work consensus algorithm, where participants compete to solve the complex mathematical problems, and the first who solves the problem gets a new block to the blockchain.

Name:Hanliang Jiang
NetID:hj33

Problem 2:

(i) Yes. As for all conflicting pairs, which is a, b and c, the order are the same, which is (T1, T2) for all.

(ii) I do not agree. For example:

```
write(a, caz, T1);
write(a, bar, T2);
read(b, T1);
read(b, T2);
write(c, baz, T2);
write(c, foo, T1);
```

the conflicting operations on a is (T1, T2) while on c is (T2, T1). Therefore, it is not serially equivalent.

(iii) All of 6 possible interleavings are serially equivalent. The only conflicting pairs is write(a, caz, T1) versus write(a, bar, T2), and the order can be either (T1,T2) or (T2,T1), and both cases are serially equivalent as there’s only one type of tag.

Name:Hanliang Jiang
NetID:hj33

Problem 3:

a. Deadlock could happen.
   Let's say T1 wants to access a,b,c in order, and T2 wants b,a,c in order, so T1 holds a and T2 holds b and they wait for each other.
b. Deadlock would not happen in this case.
   This is because if T1 is holding a and wait for T2, and T2 is holding b and waiting for a, then T2 cannot be holding b because T2 needs to get a before it can get b, as they lock objects in lexicographically increasing order. Therefore, any deadlocks cannot logically exist.
c. In this case, deadlock cannot happen, as all objects have a fixed order, just like the last case. And the proof is the same as last case.

Name:Hanliang Jiang
NetID:hj33

Problem 4:

The serial equivalence is not satisfied in this case.
Counter example: T1 locks a, T2 locks b, then T1 writes a, T1 releases a, then T2 writes b, and T2 releases b. After that, T1 locks b and T2 locks a, then T2 writes a and releases a, then T1 writes b and releases b.
In this case, a and b both have conflict access, but the tag for a is (T1, T2) and the tag for b is (T2, T1), and it undermines serial equivalence.

Name:Hanliang Jiang
NetID:hj33

Problem 5:

(a) Both 1 and 2 are cpu dominant jobs,
Limitations are: $x+2y<20$ and $x=2y$.
$X=10$ and $y=5$
Therefore, we assign job 1 10 CPUs and 10 RAM, assign job 2 10 CPUs and 10 RAM.

(b) 1 and 2 has both dominants,
Limitations:
$X+y<10$ and $x=y$
$X=5$ and $y=5$
Job 1 and job 2 are both assigned 10 CPUs and 20 RAM

(c) 1 is CPU dominant job and 2 has both dominants,
Limitations:
$2x+4y<20$ and $x+8y<40$ and $x=2y$
$Y<2.5$, but as tasks can only be integers,
$y=2$ and $x=4$,
job 1 is assigned 8 CPUs and 4 RAM, job 2 is assigned 8 CPUs and 16 RAM

(d) 1 is RAM dominant and 2 is CPU dominant.
Limitations:
$2x+6y<20$ and $8x+2y<40$ and $2x=3y$
$Y<20/7$, $y=2$ and $x=3$
job 1 is assigned 6 CPUs and 24 RAM, job 2 is assigned 12 CPUs and 4 RAM

Name:Hanliang Jiang
NetID:hj33

Problem 6:

a. Leaf nodes all report their IDs to their parents, and each intermediate node collects one highest ID and one lowest ID from its children nodes, and compares all the highest to find the highest ID among all that collected, and find the lowest among all. The node then passes the highest and lowest ID it selects to its parent node. The root finally selects one with highest ID and one with lowest ID, which is the result.
b. Leaf nodes all report their speeds and IDs to parent nodes. Each intermediate node collects the five fastest nodes from its children nodes(if less than 5 in total, report them all), and select 5 nodes with highest speed, and send the IDs and speeds of these five nodes to its parent node. The root finally selects 5 nodes with highest speeds, which is the result.
c. Leaf nodes all report 1 to parent nodes. Each intermediate node sums up all the numbers collected, and report the number to parent node. The root finally get the count of all whales, which is the result.
d. Leaf nodes each report {speed, 1} to their parents. Each intermediate node sum up all pairs' second number as num, and sum up all pairs' first number as total speed, and report {total speed, num} to its parent. The root do the same thing and calculate (total speed/num) as the final result.
e. Leaf nodes each report their speed to parents. Each intermediate node collect all the data from children and sort the data, and report the sorted array to its parent. The root do the same thing as intermediate node and find the sizeof(array)*3/4 element of array as the result.

Name:Hanliang Jiang
NetID:hj33

Problem 8:

(a) Set up is WRONG! If P4 is the owner in a write mode, P2 cannot have any copy! Let's say P2 doesn't have a copy. P3 asks P4 to invalidate its copy of pages, and P3 fetches P4's copy and uses the latest copy of P4 and marks it as (W), and P3 becomes the owner and do the write.
(b) Set up is WRONG! If P3 is the owner with a read mode, then P1 and P2 can only have read mode. Let's say P1 and P2 have read mode. P3 uses multicast to ask P2 and P1 to invalidate their copies of pages, and P3 marks the page as (W) and does the write.
(c) P3 uses multicast to ask P2 and P1 to invalidate their copies of pages, and P3 marks the page as (W) and does the write.
(d) Set up is WRONG! If P2 is the owner in a write mode, P1 and P3 cannot have any copy! Let's say P1 and P3 doesn't have a copy. P3 asks P2 to invalidate its copy of pages, and P3 fetches P2's copy and uses the latest copy of P2 and marks it as (W), and P3 becomes the owner and do the write.
(e) P3 asks P4 to invalidate its copy of pages, and P3 fetches P4's copy and uses the latest copy of P4 and marks it as (W), and P3 becomes the owner and do the write.
(f) Set up is WRONG! Only one of them can be holding a page in write mode. Let's say P3 is the owner in write mode, and P3 just needs to write to the cache.
(g) P3 asks P1 to invalidate its copy of pages, and P3 fetches P1's copy and uses the latest copy of P1 and marks it as (W), and P3 becomes the owner and do the write.
(h) P3 just needs to write to the cache.
(i) P3 uses multicast to ask P4 and P5 to invalidate their copies of pages, and P3 marks the page as (W) and becomes the owner and does the write.

Name:Hanliang Jiang
NetID:hj33

Problem 10:
Barbara Liskov is a woman computer scientist, and she has great contributions to distributed systems and distributed computing. She invented argus, a programming language that supports distributed programs. Argus supports guardians, which are objects containing both data and processes, and actions, which are atomic operations to be either committed or aborted. Barbara Liskov also has contributions to Byzantine fault tolerance, which focuses on the ability to function of a distributed system under the conditions of faulty and malicious components.

References: Barbara Liskov - Wikipedia, Byzantine fault - Wikipedia, Argus - Wikipedia