1.  
   a. Total = 23. 
      Since concurrent enqueues are disallowed, Quorum(enqueue) > Total/2. 
      Since concurrent dequeues are disallowed, Quorum(dequeue) > Total/2. 
      Since at most 3 peek are allowed, Quorum(peek) > Total/4 
      Since dequeue and enqueue are not allowed, Quorum(dequeue) + Quorum(peek) > Total 
      Hence, a possible assignment is: 
         Quorum(enqueue) = 12, Quorum(dequeue) = 18, Quorum(peek) = 6 
   
   b. With 40% enqueue, 40% dequeue and 20% peek, we must make dequeue more efficient. Quorum(enqueue) = 12, Quorum(dequeue) = 12, Quorum(peek) = 12 

2. If the coordinator cannot fail, then there is always a process which can decide. Hence, any alive cohort can contact the coordinator to know that the decision was. Therefore, it will be non-blocking. Note that all scenarios considered for non-blocking involved the coordinator failing. 

3. We must show that the channel states are correctly recorded. Let m be a message sent by i to j before it records its local state, but this message is not received by j before it records its local state. This message must be recorded as part of the channel state. Since each message takes D time, this message will be received by j before time T + D. As per the algorithm, this will be recorded as part of the channel state. 
   Similarly, let m be a message sent by i to j after it records its local state. This message must not be recorded as part of the channel state. Since each message takes D time, this message will be received by j after time T + D. As per the algorithm, this will not be recorded as part of the channel state. 
   Hence, the algorithm is correct. 

4. 
   a. $r_1(x)$ will succeed and return 0. $r_3(y)$ will succeed and return 0. $r_2(x)$ will succeed and return 0. $w_3(z)2$ will succeed and write 2. $w_1(z)5$ wants to write but the write timestamp of z is higher. As per the rules, this operation will be skilled. $w_2(y)3$ wants to write y but read timestamp of y is already 3. Hence, this operation is abort and T2 will abort. Similarly, T1 is also abort. Finally, $r_3(y)$ will succeed and return 0. 
   
   b. With two-phase locking:
      \[\begin{align*}
      RL(x) & r1(x) 0 & WL(z) & w1(z)5 & WL(y) & w1(y)4 & UL(y) & UL(z) & UL(x) \\
      RL(x) & r2(x)0 & & & & & & & \\
      RL(y) & r3(y)0 & WL(z) & w3(z)2 & & & & & \\
      \end{align*}\] 
      \[\begin{align*}
      RL(y) & r1(y)0 & WL(z) & w1(z)5 & WL(y) & w1(y)4 & UL(y) & UL(z) & UL(x) \\
      RL(x) & r2(x)0 & & & & & & & \\
      RL(y) & r3(y)0 & WL(z) & w3(z)2 & & & & & \\
      \end{align*}\]
5a.
   i. Since no one writes to array A, transactions of type T1 will never abort.
   ii. Read operations on A will not cause any abort. Since there are no read
       operations on B, write operation on B will also not abort. Hence, this
       statement is false.
   iii. Since T3 only write to locations in B, a transaction of this type cannot abort.
       Note that abort only happens due to read and no transaction reads a
       location in B.

5b.
   i. Transaction of type T1 only want to acquire read locks. Since no one writes
       to location in A, a transaction of type T1 will not have to wait for a lock.

5c. To simply two-phase lock, we can eliminate read locks on A. This will
    eliminate the overhead of acquiring and release read locks.