1. Give an example where an unreliable failure detector produces a suspected value when the system is actually functioning correctly.

   An unreliable failure detector may produce one of two values to identify a process: unsuspected or suspected. The result of unsuspected signifies the detector received evidence suggesting that the process has not failed. And the result of suspected signifies that the detector received evidence suggesting that the process may have failed.

   But even if the unreliable failure detector produces a suspected value, the system may not fail. If the detector receives no message from the process for more than a nominal maximum length of time, it gives a Suspected signal, but the suspicion may be misplaced. For example, the process could be functioning correctly but be on the other side of a network partition; or it could be running more slowly than expected; or the packages sent from the process might be corrupted or missed.

2. Give a formula for the maximum throughput of a mutual exclusion system in terms of the synchronization delay.

   The maximum throughput is the amount of requests in a critical section, which can be calculated by:
   The maximum throughput = 1 / (synchronization delay + minimum time spent in a critical section)

3. Adapt the central server algorithm for mutual exclusion to handle the crash failure of any client (in any state), assuming that the server is correct and given a reliable failure detector. Comment on whether the resultant system is fault-tolerant. What would happen if a client that possesses the token is wrongly suspected to have failed?

   The central server grants processes permission to enter the critical section. To enter a critical section, a process sends a request message to the server. If no other process has the token at the time of the request, the server replies immediately, granting the token. Otherwise, the server does not reply, but queues the request. When a process exits the critical section, it sends a message to the server and gives back the token.

   However, when a process fails, the resultant system is not fault-tolerant. The reliable failure detector could send the wrong messages. For example, after a process sends back the message to give back the token, the process fails to stop, and the server gives the token to another process, then the two processes might run concurrently. Also, if a process crashes while it holds the token, the data accessed by the process might be prone to corruption.

4. Even without a deadlock, a poor algorithm might lead to starvation. Give an example of a system which leads to starvation.

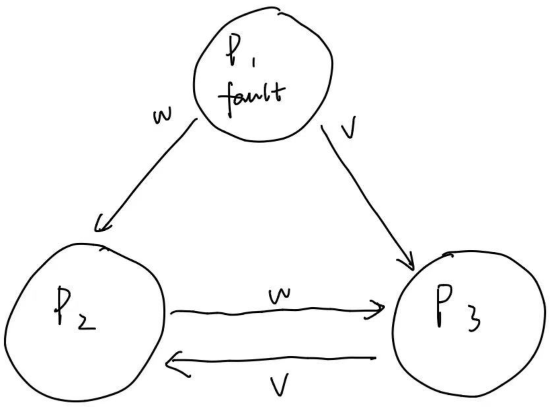
If a process keeps consuming resources, then it will lead to starvation. For example, in the following codes:

```
while (true) {
        some codes
}
```

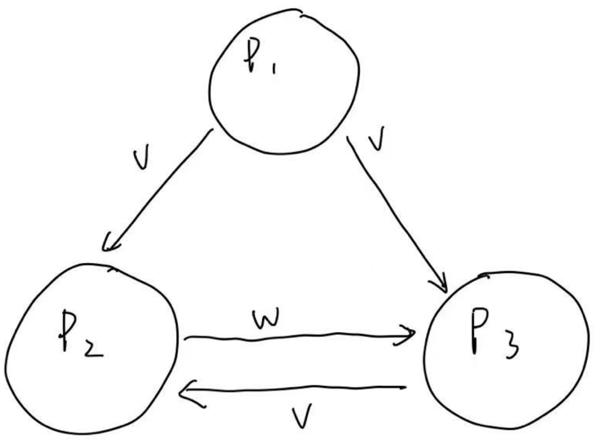
The program might have some logic to jump out of the loop inside the while loop, but if it fails to fulfill the specific condition, it will keep running till it starves the system.

5. Show that Byzantine agreement can be reached for three generals, with one of them faulty, if the generals digitally sign their messages.

   If processes sign their messages, then a faulty process is limited in the harm it can do. And it cannot make a false claim about the values that a correct process has sent to it.



Assume p1 is faulty, so p1 sends random messages to p2 and p3. From the perspective of p2, p3 receives v and sends v; p2 receives w and sends w; but p1 sends w and v, so p1 is faulty. From the perspective of p3, p3 receives v and sends v; p2 receives w and sends w; but p1 sends w and v, so p1 is faulty. p1 is faulty, so it might create any judgment, assume it detects that p1, p2 and p3 are all normal. Since there are two results of p1 failing and one result of p1 not failing, we can conclude that p1 is failing.



Assume p2 is faulty, so p2 sends random messages to p3. From the perspective of p1, it sends v to p2 and p3, p2 receives v but sends w, p3 receives v and sends v, so p2 is faulty. From the perspective of p3, p3 receives v and sends v, p2 receives v but sends w, so p2 is faulty. p2 is faulty, so it might create any judgment, assume it detects that p1, p2 and p3 are all normal. Since there are two results of p2 failing and one result of p2 not failing, we can conclude that p2 is failing.

6.

1) p1 records its state, and sends marker to p2 and p3, the channel c21 and c31 starts recording
2) p2 receives the marker from p1, it is the first marker p2 receives, so p2 records its state c12 as an empty state{}, and p2 sends marker to p1 and p3, the channel c12 and c32 start recording
3) p1 receives the marker from p2, records c21 as {A}
4) p3 receives the marker from p2, it is the first marker p3 receives, so p3 records its tate c23 as an empty state {}, and p3 sends marker to p1 and p2, the channel c13 starts recording
5) p3 receives the marker from p1, and records the state c13 as {}
6) p1 receives the marker from p3, and records the state c31 as {D}
7) p2 receives the marker from p3, and records the state c32 as {F}

c12={}
c21={A}
c23={}
c31={D}
c32={F}
c13={}

7. Explain in your own words, the Byzantine generals problem as it relates to a sensor network with three nodes. Why is this an impossible problem to solve in case of asynchronous system.

Messages in asynchronous systems may be delayed arbitrarily long, but will eventually be delivered, so we cannot set a fixed upper bounds for transmission. And since there are no upper bounds for transmission, we cannot find a reliable failure detector in asynchronous systems with crash failures.

Since we cannot detect failures in asynchronous systems, we cannot reach a consensus regarding which component has failed. For example, there are three nodes in a sensor network, and we set an averaged maximum transmission time as the upper bounds. When the network is slow or there are too many tasks in the queue, each node could detect failure from other nodes, and when there are more than half signals hinting one node has failed, we draw a conclusion that this node failed. But it could be wrong since the messages might arrive after that.