Common type of bugs

- Atomicity violation: Code region was intended to be atomic but not enforced during execution
- Ordering Violation: Order of execution is not enforced but it is rather assumed
- Deadlocks: Cyclic dependency between threads

Conditions for Deadlock

- Following four conditions to be met for deadlock to happen
  - Mutual Exclusion
  - Hold and Wait
  - No preemption
  - Circular wait
- It is to be noted that if the above conditions are met, deadlock could happen but it is not guaranteed to happen. This is due to the fact that scheduling of threads is not deterministic.

Deadlock prevention

- If any of the four conditions mentioned above could be avoided, then deadlock could be prevented before happening.
- No mutual exclusion
  - Allow sharable resources to be used by multiple threads at the same time. e.g. reading a shared region
  - Make use of wait-free synchronization but these primitives are not easy to use
- No hold and wait
  - Grab all locks at once but this assumes that the thread know the locks that needs to be acquired beforehand.
- Resource preemption
  - Try acquiring the lock but if it fails then release all the locks acquired before. Issue of livelock is possible.
- Prevent Circular wait
  - Follow some lock ordering to avoid circular dependency

Deadlock Avoidance

- Assuming we know the requirements of locking on every thread, the scheduler could schedule threads appropriately.
- This is not really possible in a larger systems (like regular OS) but this could be possible in small-scale system

Detect and Recover

- If deadlock occurrence is rare, then handle it only when the issue happens
- Detection should check for all waiting threads and the lock on which they are waiting on.
- If any cyclic chain is found, remove a link in the chain to allow progress.