Know the definitions!

- What is interference?
  - Destructive update, caused by the arbitrary interleaving of read and write actions on shared objects

- What is a monitor?
  - Private data + mutually exclusive access through monitor procedures; it supports condition synchronisation

- What is a semaphore?
  - Mechanism for synchronisation - supports up() and down(), is initialised to the number of shared resources, and its FSP is...

- What's a mutex?
  - A semaphore initialised to 1 (i.e., a single shared resource)

- What are the conditions for a deadlock?
  - Serial resources, incremental acquisition, non preemption, wait-for cycle

- Readers-writers problem and different solutions?
  - 7.5

Know the definitions!

- What is a safety property?
  - Nothing bad ever happens

- What is a liveness property?
  - Something good eventually happens

- What is a progress property?
  - At each state an action will eventually be executed

- What is starvation?
  - Lack of progress for an action

- What is fair choice?
  - Choice over a set of transitions executed infinitely often → every transition in the set will be executed infinitely often

- What is a terminal set of states?
  - Every state is reachable from any other state and no transition to a state outside the set

- How do we check safety?
  - property process - describe good behaviour

Definitions!

- What is a sequential program?
  - One thread of control

- What is a concurrent program?
  - Multiple threads of control

- Why do we use concurrent programming?
  - Performance, throughput, responsiveness, models reality

- What types of concurrent programs do we have?
  - Synchronous & asynchronous

- What is a synchronous program?
  - Processes perform actions in simultaneous execution steps

- What is an asynchronous program?
  - Processes perform actions at arbitrary relative speeds

- How do processes communicate?
  - Shared actions

- What is a critical region?
  - Part of code which accesses shared objects
Know the definitions!

- **How do we check progress?**
  - **progress** set of actions - one of these actions should be executed infinitely often.
  - We find the terminal sets of states and check whether one of the actions of each progress set is present in each of them.

- **What do we check safety properties on?**
  - Subsystem they refer to, by composing them with the property.

- **Why?**
  - They're compositional - no violation at the subsystem then no violation when the subsystem gets composed with other subsystems.

- **What do we check progress properties on?**
  - Whole model

- **Why?**
  - Not compositional - 8.1.5 for explanation.

- **How do we stress test a model?**
  - Use priorities (judiciously...)

Know FSP & Java!

- Need to understand and **be able to write** both FSP & Java.
- Make sure you understand the syntax of FSP.
- Must be able to draw the LTS of an FSP!
- Must be able to write the FSP of a LTS!
- Must be able to compose different LTSs!
- Make sure you understand how indexing of locally defined processes works and what the notations \( \{a, b\} : P \) and \( \{a, b\} : : Q \) mean.
- Make sure you understand how structure diagrams relate to the corresponding FSP and in particular how different kinds of links between components correspond to different kinds of process/action (re-)labellings.
- Understand how you can use indexed processes to model local variable for these processes.
- You should understand the author's approach to modelling as a way to develop more reliable concurrent programs. Chapter 8 and its cruise control example can help a lot.

More Revision Tips

- **Find the definitions in the book**
- **Understand them**
- **Learn them**
- **Keep it short when answering**
  - A couple of lines/sentences, not a whole page/diagrams/etc. (unless explicitly asked to provide these).
  - Long answers show that you're unsure and trying to cover it up...
  - Plus they increase the probability of making an error.
  - Plus they reduce the time you've got to do the rest of the exam.

- **Java:** in the exam you may be asked to write short sections of code, or to give variable and method definitions.
- Need to know about threads and all their mechanisms.
- Look carefully at (and test with LTSAs!!!) all examples, both in FSP & Java, in the book.
  - Try to change them and see what happens - *e.g.*, why these alphabet extensions?
  - That's the only way to really understand them...
- Fill in any lab exercises you didn't complete during the term: much the best way to revise is to **exercise** your knowledge; the exam will be more a test of understanding rather than memory.
  - There are further exercises at the end of chapters.
- **Ask questions on Moodle for extra help!**