Q1. FSM, Concurrent Models and ACM

* Components of the model: states, transitions, triggers, actions. Extensions to the model: auxiliary variables. Applications of the FSM model.
* Formal definition of FSD.
* Mealy and Moore types
* Petri Nets as an extension of FSM; the idea of concurrency as opposed to an order; concurrency is more than parallelism.
* Petri Nets: formal model, semantics.
* Reachability graphs: how they are derived from a Petri net
* REMOVED. Timing Diagrams, STG an State Graphs: STG and SG are effectively the same Petri nets and reachability graphs having additional labels, which allow their interpretation as a specification of a digital interface.
* ACM - motivation and taxonomy.
* ACM - Petri net models for all taxonomy types when using a single atomic register
* REMOVED. Synchronisation probem - Lamport's approach to the famous problem of Buridan. The idea is to use the initial conditions as a variable and the time as a parameter. Review the example of the rail road crossing.
* REMOVED. Formulate the Buridan's law. Extend the Buridan's problem to the decision making process in a circuit sampling the input from an end switch. What are the initial conditions in this example? The analysis based on the circuit theory from the lecture notes is not included, but the resultant waveforms may be useful for the explanation of your ideas.
* Have a general idea of operation of (a) the two-flop synchroniser and (b) the asynchronous arbiter. Illustrate these with waveforms and STG for the arbiter.

Q2. Software implementations of FSM, concurrent systems and ACM

* Software implementations of FSM: GoTo-Label, Switch-Case and FST. The code must be written in a pseudocode whose syntax resembles C. Any insignificant syntax errors are not penalised. You are free to "invent" your own instructions or functions as long as they are sufficiently explained. For example, if you can't remember the syntax for the static initialisation of a two-dimensional array for FST, then just draw the corresponding 2D table.
* Solve simple problems. For example, given a simple textual description, design an FSM model, then implement it as a programme in one of the above styles of coding. The models at the exam will not be more complex than a traffic light or a lift in a three storey building, etc.
* Processes: fork, exec, data communication with the static (mkfifo) of dynamic ("|") pipelines
* Threads: their difference from processes, create and join functions, critical sections, mutexes, conditional variables, experiment of fairness and its explanation with Petri nets.
* Solve a problem: given a Petri net derive the state graph; given a multithreaded programme derive a Petri net; given a Petri net derive a programme in pseudocode; given some code determine if an error is possible due to simultaneous access to the shared resource (e.g. Two functions A and B access the same memory slot - is an error possible due to these functions doing it at the same time?). The code will include creating and joining threads, mutexes and conditional variables.
* Solve a problem: given a set of timing diagrams derive an STG and then the corresponding state graph. Are there any race conditions than may cause enabling and then disabling a transition without firing, which is an indication of a possible glitch (malformed pulse). In STG/Petri nets this is related to the arbitrated choice or free choice structures (see the lecture material). In state graphs one can find a trace where a signal becomes "excited" (can be changed in the next state), but then the excitation stops if a different transition is executed.
* ACM - pseudocode as in the lecture notes. Read the recommended papers for better understanding.

   
Q3. Scheduling and Schedulers.

* Taxonomy
* Hard and soft real-time, the role of deadlines
* Metrics
* Task graphs, e.g. draw the task graph defined as a set of parameters. Determine some other parameters, e.g. relative release jitter for some particular task. See the lecture notes for the list of exercises from the book.
* Proofs of the theorems for RM and periodic EDF, as in the lecture notes.