Some solutions for practise test, AND some important points to note

Q1 and Q2 you should be able to solve now.

Q3.
P0 = (ABCDEGF)
P1 = (ABCDE)(FG) by observing the distinguishing outputs

Now 0-successors:
A -> F
B -> G
C -> B
D -> C
E -> D
FG -> EE

1-successors:
A -> B
B -> A
C -> C
D -> B
E -> A
FG -> FG

1 successors don’t give any new info. Bit 0-successors do:
(AB) -> (FG)
(CDE) -> (BCD)
So AB can be distinguished from CDE

P2 = (AB)(CDE)(FG)

Now C’s successors are in a different partition than DE’s successors.

P3 = (AB)(C)(DE)(FG)

Now 0-succ of DE -> CD which are also different, so

P4 = (AB)(C)(D)(E)(FG) and thats it.

Note, if the original machine’s reset state is A (or) B, then the minimized machine’s reset state has to be (AB). Otherwise, say if the minimized machine’s reset state is (C), then the two machines will have a different response for SOME input sequence. So, equivalence of reset states is also a necessary condition for the whole machine’s equivalence.
For the other Machine:

\[
P0 = (ABCDE) \\
P1 = (ABE)(CD) \\
P2 = (A)(BE)(CD) \\
P3 = (A)(BE)(C)(D) \text{ and that's it}
\]

Q6
(a) FALSE. A minterm denotes only a point, not a set of points, in the B-space. A Cube represents a set of points. If \( F(a, b, c) \) is a function, then \( a^*b^*c^* \) is a minterm. But \( a^*b \) is a cube that covers \( a^*b^*(c + c^*) \).

(b) True. That's what Q-M method does. Find a cover consisting of ALL primes. Then you can always find a minimum # of primes that cover the whole Function. And that is a minimal representation.

(c) True. A Prime cannot be contained in another larger cube, but it can be contained in the union of two or more primes. That's why we have to “select” a minimum # of primes.

(d) False. But why? Think. If I give you all the answers, then I might as well do your job (or write your exam). PS: Can you design an inverter using only XNOR gates? Can you design an OR gate using only XNOR gates and Inverters (where the Inv is also designed using XNOR). What about AND?

Other things to study:
- Look at counters, synchronous (DFF and TFF based), and asynchronous (TFF based).
- Design ckt's using MUXes and decoders (see ch 6)
- When you design FSMs, state your (realistic) assumptions on input and output signals. FSMs are often subject to interpretation.
- I've given you enough experience with DFF timing: setup, hold, prop delay. Don’t mess up.
- Study design of gated latches and edge-triggered FFs. They are important. SR, D-latch, DFFs, and TFFs.
- Other adder, mult, 2’s comp etc. also you should now know very well.

Good luck. Be cool. Be confident.