Study Questions

Chapter 0
0 What are formal methods?
1 How are formal methods helpful?

Chapter 1
1 What are binary expressions used for?
2 What are consistency and completeness?
3 What are the rules of proof?
4 How is monotonicity used in proof?
5 How is context used in proof?

Chapter 2
0 What are bunches used for?
1 What are sets used for?
2 How do bunches and sets differ?
3 What are strings used for?
4 What are lists used for?
5 How do strings and lists differ?

Chapter 3
0 What is the formal notation for “substitute a for b in c”?
1 What is a predicate? What is a relation?
2 What is the difference between function application and function composition?
3 How do you get the result of a quantifier applied to a function with a null domain?
4 What is a partial function? What is a total function?
5 What is a deterministic function? What is a nondeterministic function?

Chapter 4
0 How is computer behavior specified?
1 What is the difference between low level and high level specification?
2 How do you know whether a specification is implementable?
3 What does refinement mean?
4 What is a program?
5 What is refinement by steps? by parts? by cases?
6 What is a compiler's (or interpreter's) view of a program?
7 What is a prover's view of a program?
8 What is implementability with a time variable?
9 What is real-time?
10 What is recursive time?
11 What are the three levels of care in programming?
12 What variables and assignments must be added to a program to find its maximum space usage?
13 What variables and assignments must be added to a program to find its average space usage?
14 What is an assertion?
15 How could you find the initial conditions under which execution of a program would result in a satisfactory final condition?
16 What is an invariant?

Chapter 5
0 What kind of quantification is variable declaration?

1 What problem does array element assignment cause, and how is it solved?
2 How do you prove properties of while-loops?
3 How do you prove properties of for-loops?
4 Can the time variable be used in an assignment to another variable?
5 What are assertions used for? Do they help verification?
6 What are side-effects used for? Do they help verification?
7 Which is better for modularity: value parameters, or variable parameters?
8 How do you find the average value of an expression whose variables have probability distributions?
9 How do you write a probabilistic specification?
10 How do you handle a random number function formally?
11 How do you write a functional specification?
12 What is refinement between functional specifications?

Chapter 6
0 What information do you get from a construction axiom?
1 What information do you get from an induction axiom?
2 What bunch cannot be defined by construction and induction?
3 How can you find out what is defined by construction and induction?
4 Does it always work?
5 Can programs be defined by construction and induction?

Chapter 7
0 Why do you want a theory for a data structure, rather than just an implementation?
1 Why might you want a strong theory? Why might you want a weak theory?
2 How do you prove that an implementation of a data structure is correct?
3 What's the difference between a data-theory and a program-theory?
4 What's the difference between user's variables and implementer's variables?
5 What is a data transformer?
6 How do you use it?
7 What happens if you make a bad choice of data transformer?

Chapter 8
0 How do you partition the variables for a concurrent composition?
1 In a process, what information is available about other processes?
2 What is the execution time of a sequential composition? of a concurrent composition?
3 When can sequential programs become concurrent processes?
4 When is a buffer useful? How big a buffer?
5 How can you synchronize two processes at their mid points?

Chapter 9
0 What are shared, interactive, and boundary variables?
1 When are interactive variables useful?
2 How do you build a shared variable?
3 What programming problems are caused by shared variables?
4 What are the components of a communication channel?
5 Which gives processes more information about each other: shared variables, or communication channels?
6 What's a deadlock?
7 How can you tell if a computation can get into a deadlock?
8 How can you program dynamic process generation?
9 How could you build a logic-checker that works like syntax-checkers and type-checkers do now?