By the end of the course, students can…	1. Apply the formal systems we discussed to model computational systems (like programs and circuits), including reasoning about them, proving relevant properties, and communicating about them clearly and precisely with fellow Computer Scientists. Learn and apply new formalisms, specifically be able to connect between features and conclusions in the formal and informal (English language, sketch-based, pseudo-code, etc.) representations.	2. Justify the behaviour and correctness of some algorithms (e.g. at the level of selection sort and recursive binary searc or quicksort), but especially for algorithms with singly and doubly nested loops in order to prove them correct or bound their running time.	3. Translate easily among English language, simple formal representations (i.e., propositional and shallowly nested predicate logic statements), and closely related equivalent formal representations (in order to identify alternate methods to solve or simplify a variety of problems, such as writing conditionals, as you work with them). Write proofs for simple theorems by translating the theorem into first-order logic, decomposing the statement into its components, and then using the proof techniques discussed in class (direct proofs, indirect proofs by contrapositive, indirect proofs by contradiction, proofs by weak and strong mathematical induction).	4. Read a proof, and justify why each step of the proof is correct.	5. Create regular expressions and DFAs to solve problems that are important to them in programming.
Propositional Logic and Circuits (3)	C		A, B		
Proofs (4)	(G)?	(F)?	D, **E,** F	G	
Arithmetic Circuits (2)	H, I				
Sets and functions (2)	J		K		
Finite Automata (3)	L, M, N				L, M
Induction (3)		O, P	O	Q	
Relations (1)	R		R		

Topic	ID	Assessed in?	Learning Goals Students can…
Propositional Logic and Circuits	A	Implicitly assessed with B, *should* be assessed on a quiz, assignment	express simple natural language statements using propositional logic.
	B	midterm (sometimes on a quiz, but too long)	distinguish between statements that express the same information about the world versus statements that don't using logical equivalences.
	C	Lab(1-2), quiz or midterm, sometimes assignment	translate back and forth between propositional logic statement and circuits that assesses the truth or falsehood of those statements.
Proofs	D	F6a, midterm, assignments (with variety of domains), quiz	express natural language statements which require the use of predicate logic to describe, for example, the result of algorithms that use loops.
	E	Assignment, sometimes quiz	make statements about the relationships between properties of various objects (e.g. every candidate got votes from at least three people in every province).
	F	F5, F6b, F7b, F9, quizzes, assignments, midterm	create simple direct and indirect proofs, to be able to prove the correctness of operations that can be performed in programs. As another example, supports the development of data type representations (e.g. rational numbers).
	G	Not directly assessed (now one on quiz and one assignment), maybe occasionally on an assignment. Suggestion, use the web.	evaluate when a proof fails to satisfy as a communication between people – that is identify inaccuracies or missing steps in proofs.
Arithmetic Circuits	H	F1a, F1b, labs a lot, lightly on assignment	describe how the arithmetic operations of the computer break down into simpler logical operations as this is understanding one step of the layered structure of computers.
	I	F1b?, F3a, F3b, F3c, lab, breakdown not assessed otherwise	recognize why the numerical systems that we work with on computers behave the way they do, especially in cases where they break down such as floating point representation being inaccurate, overflow, and limitations of integral numerical types (longs, ints, etc.).
Sets and Functions	J1	F2a (simpler), F2b, F7a, F7b, not	apply previously developed formalism to proofs about sets and functions as applied in Java collection classes and in databases.
	J2	really the application to Java or	give examples of function that have certain properties and vice versa (e.g. injective, surjective, bijective).
	K	Continue to do questions like D/E and they understand better. Assignments, quizzes	more precisely explain the meaning of quantified statements. (elaboration of D/E)
Finite Automata	L	F11a, lab (adding a new one), assignment,	model and solve real world problems such as control circuits (traffic lights), matching problems, validating input, and (in the abstract) modeling the capabilities of a computer using real circuits/DFAs.
	M	F11b, assignment, quiz, lab	Students can create regular expressions which produce DFAs to solve problems that are important to them in programming.
Induction	O	F8(not prog), F10b but easier, too hard to assess, not convinced that we have a simple enough problem that they can do. Save for 221., assignment (a lot), quiz	prove things about programs that the use loops and recursion.
	P	F8(not prog)	justify the correctness of a reasonably complex recursive algorithm (like quicksort or mergesort). An example of O.
	Q	F8(not prog), F10a, talk a lot about in class, but not on assignment, the application can just be done mechanically (NOT ASSESSED BEFORE FINAL)	be able to list out the exhaustive steps from a proof that should prove that -- given a property that they want to prove and given any specific value to prove that property at.
Relations	R	Sometimes we get to it and sometimes we don't.	prove that a relation is symmetric, transitive or reflexive.