Study questions on Discrete Structures for Computer Science

The intention in providing these questions is to show the student what sorts of fact and problems we address in this course.

Most of the multiple-choice questions are factual. Knowing that one can answer the questions correctly can raise your confidence in your learning. Awareness of not knowing answers of some questions can help guide your review.

Multiple-choice questions are organized by subtopic in the course plan. Questions below are intended to correspond to slides, in content and in ordering. I appreciate hearing about questions that don't correspond fully.

In certain versions of this file, answers to multiple-choice questions are supplied. Grading of all quizzes will be according to the *correct* answer, not the answer that has been provided in some list of answers. Please question any purported correct answers that you don't agree with or don't understand.

Contents

Study questions on Introduction and background

1. What this course offers

1. An example of analog representation is (a) a file stored on a computer; (b) a message sent on the Internet; (c) the sound heard from an IPod; (d) a picture in RAM; (e) a register in a processor
2. Analog is to digital as continuous is to (a) binary; (b) infinite; (c) discrete; (d) irrational; (e) none of these
3. Discrete is to continuous as (a) binary is to decimal; (b) real is to integer; (c) digital is to analog; (d) infinite is to finite; (e) none of these
4. An algorithm *lacks* which of these features? (a) computes a function; (b) is deterministic; (c) may take an unreasonably long time; (d) works in discrete steps; (e) may never end
5. Algorithm specifications presuppose (a) that input has occurred; (b) that processing has occurred; (c) that output has occurred; (d) the meaning of input; (e) that loops time out
6. Algorithms solve problems that are associated with (a) services; (b) protocols; (c) irrational numbers; (d) functions; (e) none of these
7. A function is a (a) truth value; (b) data item; (c) algorithm; (d) process; (e) mapping
8. A function may often be computed by a(n) (a) service; (b) interactive protocol; (c) multi-agent system; (d) algorithm; (e) event-driven program
9. Input to an algorithm is (a) necessarily atomic; (b) obtained before algorithm execution; (c) obtained during execution; (d) necessarily compound; (e) possibly infinite
10. An algorithm is a(n) (a) program; (b) plan; (c) structure; (d) service; (e) process
11. Discrete structures are (a) algorithms; (b) real numbers; (c) objects; (d) truth values; (e) arrays
12. Symbols are (a) analog; (b) real; (c) discrete; (d) continuous; (e) waves

3. Logic and proof techniques

1. $\wedge$ denotes (a) set membership; (b) union; (c) AND; (d) a relation between sets; (e) negation
2. $\neg$ denotes (a) set membership; (b) union; (c) AND; (d) a relation between sets; (e) logical negation
3. Logic manipulates (a) numbers; (b) algorithms; (c) truth values; (d) sound; (e) strings
4. $\vee$ denotes (a) set membership; (b) union; (c) AND; (d) OR; (e) implication
5. $\Rightarrow$ denotes (a) set membership; (b) union; (c) AND; (d) OR; (e) implication
6. A *logic is* (a) a language; (b) a rule; (c) a set of truth values; (d) a set of numeric values; (e) none of these
7. Logic manipulates (a) strings; (b) numbers; (c) truth values; (d) programs; (e) objects
8. If p = false, q = false, and r = true, then which is true? (a) $\neg p \wedge (q \wedge r)$; (b) $\neg p \vee (q \wedge r)$; (c) $(\neg p \vee q) \wedge r$; (d) $\neg p \vee (q \wedge r)$; (e) $p \vee \neg(q \wedge r)$
9. (T-F) If we live on Pluto, then cats have wings.
10. (T-F) If airplanes fly, then 1 + 1 = 2.
11. (T-F) If the earth is flat, then 1 + 1 = 2.
12. (T-F) If the earth is round, then 1 + 1 = 3.
13. (T-F) If trees have ears, then dogs have wings.
14. (T-F) 2 + 2 = 4 only if 1 + 1 = 3.
15. An if-then assertion whose first clause is true is (a) never true; (b) sometimes true; (c) always true; (d) meaningless; (e) none of these
16. A rigorous demonstration of the validity of an assertion is called a(n) (a) proof; (b) argument; (c) deduction; (d) contradiction; (e) induction
17. A proof that begins by asserting a claim and proceeds to show that the claim cannot be true is by (a) induction; (b) construction; (c) contradiction; (d) prevarication; (e) none of these
18. A proof that proceeds by showing the existence of something desired is by (a) induction; (b) construction; (c) contradiction; (d) prevarication; (e) none of these
19. Proofs by contradiction (a) dismiss certain rules of logic; (b) misrepresent facts; (c) start by assuming the opposite of what is to be proven; (d) end by rejecting what is to be proven; (e) none of these
20. Induction is a(n) (a) algorithm; (b) program; (c) proof; (d) proof method; (e) definition
21. Contradiction is a(n) (a) algorithm; (b) program; (c) proof; (d) proof method; (e) definition
22. Construction is a(n) (a) algorithm; (b) program; (c) proof; (d) proof method; (e) definition
23. A proof that begins by asserting a claim and proceeds to show that the claim cannot be true is by (a) induction; (b) construction; (c) contradiction; (d) prevarication; (e) none of these

Inductive proof

1. The induction principle makes assertions about (a) infinite sets; (b) large finite sets; (c) small finite sets; (d) logical formulas; (e) programs
2. A proof that proceeds by showing that a tree with n vertices has a certain property, and then shows that adding a vertex to any tree with that property yields a tree with the same property, is (a) direct; (b) by contradiction; (c) by induction; (d) diagonal; (e) none of these
3. A proof that shows that a certain property holds for all natural numbers is by (a) induction; (b) construction; (c) contradiction; (d) prevarication; (e) none of these
4. The principle of mathematical induction states that if zero is in a set A, and if membership of any value x in A implies that $(x + 1)$ is in A, then (a) A is all natural numbers; (b) the proof is invalid; (c) A is the null set; (d) A is x; (e) A is $\{x\}$
5. In an inductive proof, showing that $P(0)$ is true is (a) the base step; (b) the inductive step; (c) unnecessary; (d) sufficient to prove $P(x + 1)$; (e) sufficient to prove $P(x)$ for all x
6. In an inductive proof, showing that $P(x)$ implies $P(x + 1)$ is (a) the base step; (b) the inductive step; (c) unnecessary; (d) sufficient to prove $P(x)$ for some x; (e) sufficient to prove $P(x)$ for all x
7. In an inductive proof, showing that $P(0)$ is true, and that $P(x)$ implies $P(x + 1)$, is (a) the base step; (b) the inductive step; (c) unnecessary; (d) sufficient to prove $P(x)$ for some x; (e) sufficient to prove $P(x)$ for all x

8. An inductive proof with graphs might proceed by (a) showing a contradiction; (b) showing a counter-example; (c) considering all graphs one by one; (d) starting with some simple graph and adding one vertex or edge; (e) none of these
9. The base step in an inductive proof might (a) show that $P(0)$ is true, and that $P(x)$ implies $P(x + 1)$; (b) show that $P(0)$ is true; (c) show that that $P(x)$ implies $P(x + 1)$; (d) give a counterexample; (e) assume the opposite of what is to be proven
10. The inductive step in an inductive proof might (a) show that $P(0)$ is true, and that $P(x)$ implies $P(x + 1)$; (b) show that $P(0)$ is true; (c) show that that $P(x)$ implies $P(x + 1)$; (d) give a counterexample; (e) assume the opposite of what is to be proven
11. An inductive proof might consist of (a) showing that $P(0)$ is true, and that $P(x)$ implies $P(x + 1)$; (b) showing that $P(0)$ is true; (c) showing that that $P(x)$ implies $P(x + 1)$; (d) giving a counterexample; (e) assuming the opposite of what is to be proven, and proving a contradiction

4. Sets, relations, and functions

1. $\subseteq$ denotes (a) set membership; (b) union; (c) conjunction; (d) a relation between sets; (e) negation
2. $\cup$ denotes (a) set membership; (b) union; (c) AND; (d) a set; (e) negation
3. $\varnothing$ denotes (a) set membership; (b) union; (c) AND; (d) a set; (e) negation
4. $\in$ denotes (a) set membership; (b) union; (c) AND; (d) a relation between sets; (e) negation
5. $\{1,2,3\} \cup \{2,4,5\} =$ (a) {}; (b) {1,2}; (c) 2; (d) {2}; (e) {1,2,3,4,5}
6. $\{1,2,3\} \cap \{2,4,5\} =$ (a) {}; (b) {1,2}; (c) 2; (d) {2}; (e) {1,2,3,4,5}
7. (T-F) $\{1, 3\} \in (\{1, 3, 5\} \cap \{1, 5\})$
8. (T-F) $\varnothing \subseteq \varnothing$
9. (T-F) $\varnothing \in \varnothing$
10. (T-F) $\varnothing \subseteq \{\varnothing\}$
11. (T-F) $\varnothing \in \{\varnothing\}$
12. {} is a subset of (a) itself only; (b) no set; (c) all sets; (d) only infinite sets; (e) none of these
13. A *relation* on set A is (a) an element of A; (b) a subset of A; (c) an element of $A \times A$; (d) a subset of $A \times A$; (e) none of these
14. A function $f : \{1,2,3\} \rightarrow \{0,1\}$ is a set of (a) integers; (b) ordered pairs; (c) sets; (d) relations; (e) none of these
15. A string is (a) a set of symbols; (b) a sequence of characters; (c) a relation; (d) a set of sequences; (e) none of these
16. The null set is a (a) member of itself; (b) member of any set; (c) subset of any set; (d) superset of any set; (e) none of these
17. The power set of A is (a) the set of all members of A; (b) a subset of A; (c) the set of subsets of A; (d) the null set; (e) an intersection
18. For all sets A (a) $A \subseteq A$; (b) $A \in A$; (c) $A \neq A$; (d) all of these; (e) none of these
19. A *relation* on set A is (a) an element of A; (b) a subset of A; (c) an element of $A \times A$; (d) a subset of $A \times A$; (e) none of these
20. The Cartesian product of two sets is a(n) (a) set of sets; (b) ordered pair; (c) set of ordered pairs; (d) subset of the two sets; (e) union of the two sets
21. $(A \times B)$ is (a) the set containing elements of A and B; (b) the set of ordered pairs of elements chosen from A and B respectively; (c) any relation of elements of A and B; (d) a function from A to B; (e) none of these
22. We may represent a Cartesian product as a (a) linear array; (b) linked list; (c) matrix; (d) tree; (e) none of these
23. A relation is *not* a (a) set of ordered pairs; (b) set of numbers; (c) subset of a Cartesian product; (d) way to express how two sets relate; (e) it is all of these
24. A function f: $\{1,2,3\} \rightarrow \{0,1\}$ is a set of (a) integers; (b) ordered pairs; (c) sets; (d) relations; (e) none of these
25. When A and B are sets, $(A \times B)$ is (a) a set of ordered pairs; (b) an arithmetic expression; (c) a sequence of values; (d) all of these; (e) none of these

Discrete-math / finite-math terminology

algorithm
analog data
binary relation
binary tree
Cartesian product
ceiling function
conjunction
construction
contradiction
database
disjunction
domain
existential quantifier
floor function
function
graph
implication
induction
integer
intersection
logic
natural number
negation
one-to-one
path
predicate
predicate logic
principle of mathematical induction
proper subset
propositional logic
range
rational number
real number
relation
relative complement
sequence
set theory
set
subset
tree
union

Multiple-choice questions on Topic 1 (Boolean algebras)

1. Propositional logic and Boolean algebras

1. An algebra is (a) a set of integers; (b) any set of values; (c) a set of values and operations on them; (d) a set of operations; (e) a set of functions
2. A Boolean algebra includes operations with the ____ property (a) transitive; (b) reflexive; (c) commutative; (d) monotonic; (e) completeness
3. A Boolean algebra includes operations with the ____ property (a) transitive; (b) reflexive; (c) associative; (d) monotonic; (e) completeness
4. A Boolean algebra includes operations with the ____ property (a) transitive; (b) reflexive; (c) distributive; (d) monotonic; (e) completeness
5. Any set *A*, plus two binary operations on *A* with the associative and other properties, is (a) the whole numbers; (b) propositional logic; (c) set theory; (d) a Boolean algebra; (e) any algebra
6. $(\wp(U),\{\cup, \cap\})$ is (a) complete; (b) inconsistent; (c) a Boolean algebra; (d) a temporal logic; (e) a set of numbers
7. A set with the identity property has an element (a) 0, s.t. $(\forall x \in A)\ x + 0 = x$; (b) 0, s.t. $(\forall x \in A)\ x \times 0 = x$; (c) 1, s.t. $(\forall x \in A)\ x + 1 = x$; (d) that is identical to some other element; (e) that is identical to all other elements
8. For algebra *A*, if $x \in A$ then x^{-1} is the ____ of x (a) identity value; (b) complement; (c) negation; (d) reciprocal; (e) none of these
9. Propositional logic is (a) complete; (b) inconsistent; (c) a Boolean algebra; (d) a temporal logic; (e) a set of numbers
10. If x is an element of a Boolean algebra, then $(x^{-1})^{-1}$ = (a) 0; (b) 1; (c) x; (d) x^{-1}; (e) not x
11. An interpretation is (a) an assignment of truth values; (b) the value of an assertion; (c) the meaning of a program; (d) a formula; (e) none of these
12. An *interpretation* of a set of formulas in predicate logic is (a) a logical inference; (b) a heuristic; (c) an assignment of truth values to variables; (d) a theorem; (e) a truth value
13. The semantics of propositional logic specify (a) numeric values; (b) results of operations; (c) rules for constructing formulas; (d) the meaning of $\in$; (e) none of these
14. $(p \rightarrow q)$ iff (a) $p \wedge q$; (b) $\neg p \vee \neg q$; (c) $\neg p \vee q$; (d) $p \vee q$; (e) $q \rightarrow p$
15. An assertion's value is (a) true; (b) a symbol; (c) a number; (d) true or false; (e) none of these
16. A truth table contains (a) variables; (b) formulas; (c) values of formulas under one interpretation; (d) values of formulas under all interpretations; (e) operations
17. If formulas ϕ and φ have the same truth table, then (a) $\phi \Leftrightarrow \varphi$; (b) $\neg\phi \vee \neg\varphi$; (c) $\phi \wedge \varphi$; (d) $\phi \rightarrow \varphi$; (e) $\phi \vee \varphi$
18. Satisfiability is ___ validity (a) weaker than; (b) equivalent to; (c) stronger than; (d) a subset of; (e) none of these
19. A sentence that is not true under any interpretation is (a) complete; (b) incomplete; (c) consistent; (d) inconsistent; (e) valid
20. A sentence that is true under all interpretation is (a) complete; (b) incomplete; (c) consistent; (d) inconsistent; (e) valid
21. A formula is satisfiable if it has a(n) ____ under which it is true (a) operation; (b) algorithm; (c) number; (d) interpretation; (e) none of these
22. Satisfiability is ___ validity (a) weaker than; (b) equivalent to; (c) stronger than; (d) a subset of; (e) none of these
23. SAT is the problem of deciding whether a formula in propositional logic (a) holds; (b) has a set of variable assignments that make it true; (c) is not a contradiction; (d) is syntactically correct; (e) is probably true
24. The sentence, $\alpha \models \beta$ (in every interpretation where α is true, β is true), is an instance of (a) entailment; (b) negation; (c) validity; (d) satisfiability; (e) falsehood
25. Inference rules maintain (a) completeness; (b) consistency; (c) validity; (d) satisfiability; (e) falsehood
26. An inference rule that never produces contradictions is (a) complete; (b) incomplete; (c) inconsistent; (d) sound; (e) useless
27. $(p \wedge (p \rightarrow q)) \rightarrow q$ is (a) false; (b) Modus Ponens; (c) inconsistent; (d) not always true; (e) none of these
28. A validity-maintaining procedure for deriving sentences in logic from other sentences is a(n) (a) proof; (b) theorem; (c) algorithm; (d) inference rule; (e) inference chain
29. *p iff q* means (a) $p \Rightarrow q \wedge q \Rightarrow p$; (b) $p \Rightarrow q \vee q \Rightarrow p$; (c) $p \Rightarrow q$ but not necessarily $q \Rightarrow p$; (d) $q \Rightarrow p$ but not necessarily $p \Rightarrow q$; (e) none of these
30. Inference is (a) commutative; (b) transitive; (c) undecidable; (d) time dependent; (e) associative
31. The property asserted by $(p \rightarrow q \wedge q \rightarrow r) \Rightarrow (p \rightarrow r)$ is (a) commutative; (b) transitive; (c) undecidable; (d) time dependent; (e) associative
32. The property asserted by $(p = q \wedge q = r) \Rightarrow (p = r)$ is (a) commutative; (b) transitive; (c) undecidable; (d) time dependent; (e) associative

2. Predicate logic

1. Quantifiers ____ variables (a) negate; (b) change; (c) bind; (d) define; (e) give values to
2. To bind a variable in an expression like Odd(x), what are used? (a) arithmetic operators; (b) logical operators; (c) quantifiers; (d) predicates; (e) negations
3. When multiple quantifiers are the same, then then the meaning of a predicate logic sentence (a) depends on order; (b) is ambiguous; (c) is independent of order; (d) is determined by arithmetic operators; (e) is determined by logical operators
4. When multiple quantifiers differ, then the meaning of a predicate logic sentence (a) depends on order; (b) is ambiguous; (c) is independent of order; (d) is determined by arithmetic operators; (e) is determined by logical operators
5. Predicate logic is a(n) (a) algorithm; (b) language of assertions; (c) language of arithmetic expressions; (d) set of symbols; (e) set of operations
6. $(\forall x)\ x = x + 1$ is (a) a numeric expression; (b) false; (c) true; (d) an assignment; (e) none of these

7. ($\exists x$) $x = x + 1$ is (a) a numeric expression; (b) false; (c) true; (d) an assignment; (e) none of these
8. Quantifiers ____ variables for meaningful use (a) give values to; (b) take values from; (c) bind; (d) assign; (e) declare
9. Predicate calculus extends propositional logic with (a) inference; (b) negation; (c) implication; (d) variables; (e) quantifiers
10. A formula in logic is valid if (a) it is true for some interpretation; (b) it is true for all interpretations; (c) it is true for no interpretation; (d) it is an axiom; (e) it is not disproven
11. A formula in logic is satisfiable if (a) it is true for some interpretation; (b) it is true for all interpretations; (c) it is true for no interpretation; (d) it is an axiom; (e) it is not disproven
12. A formula in logic is inconsistent if (a) it is true for some interpretation; (b) it is true for all interpretations; (c) it is true for no interpretation; (d) it is an axiom; (e) it is not disproven
13. Inference rules enable derivation of (a) axioms; (b) other inference rules; (c) new true assertions; (d) percepts; (e) none of these
14. Inference rules maintain (a) completeness; (b) consistency; (c) validity; (d) satisfiability; (e) falsehood
15. An inference rule that never produces contradictions is (a) complete; (b) incomplete; (c) inconsistent; (d) sound; (e) useless

3. Some proof methods

1. Existentially quantified assertions may be proven by (a) contradiction; (b) induction; (c) showing an instance; (d) diagonalization; (e) counter-example
2. Forward chaining (a) is goal driven; (b) starts with an assertion to be proven; (c) is data driven; (d) is not sound; (e) none of these
3. Backward chaining (a) is goal driven; (b) is sound; (c) generates all possible entailments; (d) applies modus ponens; (e) starts with the data at hand
4. An algorithm that determines what substitutions are needed to make two sentences match is (a) resolution; (b) inference; (c) unification; (d) contradiction; (e) nonexistent
5. Unification is (a) an algorithm for making substitutions so that two sentences match; (b) a proof method; (c) an inference rule; (d) a theorem; (e) a knowledge-representation scheme
6. Resolution proof uses (a) forward chaining; (b) contradiction; (c) abduction; (d) unification; (e) statistics

See also questions on induction in Introduction topic, subtopic 2.

4. Inductive proofs of correctness

1. Which are sufficient conditions for algorithm correctness? (a) good programming methodology; (b) customer satisfaction; (c) approval by QA; (d) output is specified function of input; (e) program always halts and output is specified function of input
2. Total correctness is partial correctness plus (a) termination; (b) proof; (c) loop invariant; (d) postcondition; (e) efficiency
3. An assertion is (a) a comment that describes what happens in an algorithm; (b) a command; (c) a claim about the state of the computation; (d) an algorithm; (e) none of these
4. The purpose of assertions in formal verification is to (a) help establish that code is correct; (b) describe what happens in a program; (c) guarantee that a program halts; (d) catch exceptions; (e) all the above
5. A loop invariant is asserted to be true (a) throughout the loop body; (b) at the beginning of every iteration of a loop; (c) is the same as the postcondition; (d) all the above; (e) none of the above
6. An assertion that is true at the start of each iteration of a loop is (a) a precondition; (b) a loop invariant; (c) a postcondition; (d) a loop exit condition; (e) none of these
7. A loop invariant asserts that (a) the precondition holds; (b) the postcondition holds; (c) a weaker version of the postcondition holds; (d) the algorithm terminates; (e) none of these
8. A postcondition (a) is asserted to be true before an algorithm executes; (b) is asserted to be true at the beginning of every iteration of a loop; (c) is asserted to be true after an algorithm executes; (d) all the above; (e) none of the above
9. A precondition is asserted to be true (a) before an algorithm executes; (b) at the beginning of every iteration of a loop; (c) after an algorithm executes; (d) all the above; (e) none of the above
10. A Hoare triple consists of (a) precondition, loop invariant, postcondition; (b) program, loop invariant, postcondition; (c) precondition, program, postcondition; (d) proof, loop invariant, program; (e) none of these
11. A Hoare triple specifies (a) loop invariant and postcondition; (b) precondition, program and postcondition; (c) program and postcondition; (d) performance requirements; (e) none of these
12. $<\phi> P <\psi>$ is a (a) precondition; (b) loop invariant; (c) postcondition; (d) Hoare triple; (e) first-order logic formula
13. In $<\phi> P <\psi>$, ϕ is a (a) precondition; (b) loop invariant; (c) postcondition; (d) Hoare triple; (e) Boolean literal
14. In $<\phi> P <\psi>$, φ is a (a) precondition; (b) loop invariant; (c) postcondition; (d) Hoare triple; (e) Boolean literal
15. In $<\phi> P <\psi>$, P is a (a) precondition; (b) loop invariant; (c) postcondition; (d) program; (e) propositional-logic formula

Terminology for Topic 1 (Boolean algebras)

algebra	constructive proof	implication	partial correctness	termination
arity	De Morgan's Laws	induction principle	postcondition	total correctness
assertion	disjunction	inductive case	precondition	transitivity
automated reasoning	entailment	inference	predicate	truth assignment
backward chaining	existential quantifier	interpretation	predicate logic	truth tables
base case	first-order logic	loop invariant	proof procedure	unification
Boolean algebra	forward chaining	model	property	universal quantifier
Boolean variable	Hoare triple	modus ponens	propositional logic	validity
complement	idempotent	modus tollens	resolution	
conjunction	identity	negation	satisfiability	

Objectives-related questions on topic 1

1.1a Describe the syntax of propositional logic (essential)

1. Describe the *literals* in propositional logic.
2. Describe the *operators* in propositional logic.
3. Describe the syntax of propositional-logic formulas.
4. What may appear in parentheses in a propositional-logic formula?

(5-10) Why is each of the following not a propositional-logic formula?

5. $p \neg q$
6. $\wedge p \ q$
7. $p \neg \wedge q$
8. $p \wedge q \neg$
9. $p \vee q \wedge$
10. $p (\vee q)$

1.1b Apply the semantics of propositional logic (essential)

Write truth tables for the following assertions:

1. $(\neg p \vee q) \wedge r$
2. $(p \vee \neg q) \wedge \neg r$
3. $(p \rightarrow q) \wedge \neg r$
4. $(p \wedge \neg q) \rightarrow r$
5. $\neg p \wedge (\neg q \vee r)$

1.1c Apply logical inference(essential)

Write simpler propositional-logic formulas, equivalent to the following, using Modus Ponens, Modus Tollens, or the definition of implication; and naming the rule you used. You may abbreviate words with their initials; e.g., "c" =" clouds".

1. $(q \rightarrow r) \wedge q$
2. $\neg p \wedge (q \rightarrow p)$
3. $(\neg r \wedge q) \wedge r$
4. $q \wedge (\neg q \vee p)$
5. Dark clouds mean it will rain; and I see dark clouds.
6. There's no class on holidays. There's class today.

1.1d Explain Boolean algebras (essential)

1. What are the features of a Boolean algebra? Discuss in relation to a logic.
2. What are the identity elements in propositional logic? Relate to operations.
3. What is the complement of *true* in propositional logic? Relate to operations.
4. What are the identity elements for two operators in propositional logic? What mathematical structure has identity elements and complements?

Defend or refute:

5. Certain basic set operations together form a Boolean algebra.
6. Propositional logic is a Boolean algebra.
7. The natural numbers form the basis for a Boolean algebra.

1.2a Use a quantifier (essential)

Use quantifiers and predicates to express the following in predicate logic.

1. Some athletes are fast.
2. All athletes are strong.
3. Some fast people are athletes.
4. All strong people are athletes.
5. Some athletes are not tall.
6. All tall athletes are strong.
7. All fast strong people are athletes.
8. Some strong people aren't athletes.

1.2b Distinguish predicate from propositional logic (essential)

1. What two features distinguish predicate logic from propositional logic?
2. Name and describe the sorts of assertions that predicate logic can express that propositional logic cannot.
3. Describe the meanings of $\forall$, $\exists$, and $P(x)$, and name the logic that supports them.
4. Describe some limitations of propositional logic and state how another logic overcomes them.
5. Describe the *quantifiers* and how they address a limitation of propositional logic.

1.3a Write a direct proof (essential)

Use direct proof to show that

1. the product of any natural number and an even natural number is even.
2. the difference between any two even natural numbers is even.
3. for any $m \geq 3$, $m^2 - 4$ is non-prime.
4. the sum of an even natural number and an odd one is odd.

5. for any integers *a*, *b*, the difference between a^2 and b^2 is an odd number.

1.3b Write a proof by construction (essential)

Prove by construction, giving the predicate being proven.

1. 24 is divisible by both 2 and 6.
2. 20 is divisible by both 4 and 5.
3. 10 is the sum of two odd numbers.
4. 13 is the sum of an even number and an odd number.
5. 22 is the sum of two even numbers.
6. There exist two consecutive numbers that add up to 17.

1.3c Write a proof by contradiction (essential)

Prove by contradiction that:

1. No largest integer exists.
2. No smallest positive real number exists.
3. The sum of two even numbers is always an even number.
4. The sum of two odd numbers is always an even number.
5. The sum of an even and an odd number is always odd.
6. The difference between an even and an odd number is odd.

1.3d Describe the principle of mathematical induction (essential)

1. Describe the two parts of an inductive proof.
2. What is the principle of mathematical induction?
3. What sorts of theorems can the principle of mathematical induction be used to prove?
4. In an inductive proof, what must be shown, other than $P(0)$?
5. Explain the role of $P(n) \Rightarrow P(n + 1)$ in some mathematical proofs.

1.3e Use induction to prove a theorem about numbers (essential)

Prove by mathematical induction that for all natural numbers greater than zero,

1. $n^2 + n = (2 + 4 + 6 + \ldots + 2n)$
2. $\sum_{k=1}^{n} k = (n^2 + n)/2$
3. $(n^3 + 2n)$ is divisible by 3
4. $1 + 6 + 11 + \ldots + (5n - 4) = (5n^2 - 3n) / 2$
5. $1 + 3 + 5 + \ldots + (2n - 1) = n^2$
6. $2^0 + 2^1 + 2^2 + \ldots + 2^n = 2^{n+1} - 1$

1.4a Explain concepts of algorithm correctness (priority)

1. What is an assertion, about the state of a repetitive process, that holds at the start of the process and helps to establish that the process spec is satisfied? How is it used?
2. What are three classes of comments that help establish that the spec of a procedure is satisfied? For each, state where the comment should appear in the code or pseudocode.
3. For an algorithm, what is the likely relationship between a *loop invariant* and a *postcondition*?
4. How are loop invariants related to induction?
5. Distinguish *partial* from *total* correctness.
6. Identify the components of $<\phi>$ P $<\psi>$ as discussed in class, and the meaning and purpose of this.

1.4b Use induction to prove an algorithm correct*

By use of preconditions, postconditions, and loop invariants, prove that the pseudocode below is correct.

1. ***Count-spaces*(*s*)**
```
> Returns number of
> spaces in string.
y ← 0
i ← 1
while i ≤ length(s) do
    if s[ i ] = ' '
        y ← y + 1
    i ← i + 1
return y
```

2. ***Search-stack* (*S, key*)**
```
> Tells whether stack S
> contains key
found ← false
while not empty(S)
    test ← Pop(S)
    if test = key
        found ← true
return found
```

3. **All-same (*A*)**
```
> Tells whether all
> elts of A are same
y ←true
for i ← 2 to |A|
    If A[ i ] = A[i – 1]
    y ←false
return y
```

4. **Quotient (*a*, *b*)**
```
> Performs integer division
y ← 0
s ← a – b
while s > 0
    s ← s – b
    y ← y + 1
return y
```

5. **Largest-to-right (*A*)**
```
> Returns A after moving the
> largest element to right.
largest ← 1
for i ← 2 to |A| do
    if A[ i ] > A [largest]
        largest ← i
A[largest] with A[|A|]
return A
```

6. **Fact (*x*)**
```
> Computes factorial:
y ← 1
i ← 1
while i < x
    y ← i × y
    i ← i + 1
return y
```

8. **Index-of-largest (A)**
```
> Returns index of the
> largest element of A
y ← 1
for i ← 1 to |A| – 1)
    if A[ i ] < A[y]
        y ← i
    i ← i + 1
return y
```

7. **Max (*A*)**
```
> Returns largest elt of A
y ← A[1]
i ← 1
while i < |A|
    if y < A[ i ]
        y ← A[ i ]
    i ← i + 1
return y
```

9. **Pow (a, b)**
```
> returns a^b
y ← a
i ← 1
while i < b
    y ← a × y
    i ← i + 1
return y
```

10. **Sum (*A*)**
```
> Computes sum of
> array elements
y ← 0
i ← 1
while i ≤ |A|
    y ← y + A[ i ]
    i ← i + 1
return y
```

11. **Product (x, y)**
```
>Performs multiplication
result ← 0
For i ← 1 to x
    result ← result + y
Return result
```

12. **Which-sort (A)**
```
for i ← size(A) down to 2 do
    A← Largest-to-right (A[1.. i ])
```
(You may assume that Largest-to-right (#5 above) is correct).

Multiple-choice questions on Topic 2 (Sets, relations)

1. Properties of sets

1. For sets A and B, $A \cap B$ = (a) A; (b) B; (c) $B \cap A$; (d) $A \cup B$; (e) $A - B$
2. For sets A and B, $(A \cap B)$ (a) $\supset A$; (b) $\subseteq A$; (c) $\supseteq B$; (d) $\subset B$; (e) $= A - B$
3. $(A \subseteq B) \wedge (B \subseteq C) \Rightarrow (A \subseteq C)$ is a(n) ___ property; (a) associative; (b) commutative; (c) identity; (d) transitive; (e) inverse
4. $(A \cap B) \cap C = A \cap (B \cap C)$ is a(n) property (a) associative; (b) commutative; (c) identity; (d) transitive; (e) inverse
5. $A \cup A^c$ = (a) U; (b) A; (c) A^c; (d) $\varnothing$; (e) none of these
6. $(A^c)^c$ = (a) U; (b) A; (c) $U - A$; (d) $\varnothing$; (e) none of these
7. $A \cap A^c$ = (a) U; (b) A; (c) $U - A$; (d) $\varnothing$; (e) none of these
8. $A \cap \varnothing$ = (a) U; (b) A; (c) $U - A$; (d) $\varnothing$; (e) none of these
9. $A \cup \varnothing$ = (a) U; (b) A; (c) $U - A$; (d) $\varnothing$; (e) none of these
10. $A \cup U$ = (a) U; (b) A; (c) $U - A$; (d) $\varnothing$; (e) none of these
11. $A \cap U$ = (a) U; (b) A; (c) $U - A$; (d) $\varnothing$; (e) none of these
12. To prove that sets A and B are equal, prove that (a) $A \subseteq B \wedge B \subseteq A$; (b) $A \subset B \wedge B \subset A$; (c) $A \subseteq B \vee B \subseteq A$; (d) $A \subset B \vee B \subset A$; (e) none of these
13. $x \in A^c$ implies (a) $x \in A$; (b) $x = A^c$; (c) $x \notin A$; (d) $A = \varnothing$; (e) none of these
14. Sets A and B are disjoint iff $A \cap B$ = (a) A; (b) B; (c) U; (d) $\varnothing$; (e) none of these
15. If $\{A_1, A_2, \ldots\}$ partitions A, then $A_1, A_2, \ldots$ (a) are the same; (b) are disjoint; (c) are in a subset relation to each other; (d) have a non-null intersection; (e) none of these

2. Relations

1. In a symmetric relation R over A, (a) $(\forall\, x \in A)\ xRx$; (b) $(\forall\, x, y \in A)\ xRy \Rightarrow yRx$; (c) $(\forall x,y,z \in A)\ xRy \wedge yRz \Rightarrow xRz$; (d) all of these; (e) none of these
2. In a transitive relation R over A, (a) $(\forall\, x \in A)\ xRx$; (b) $(\forall\, x, y \in A)\ xRy \Rightarrow yRx$; (c) $(\forall x,y,z \in A)\ xRy \wedge yRz \Rightarrow xRz$; (d) all of these; (e) none of these
3. In a reflexive relation R over A, (a) $(\forall\, x \in A)\ xRx$; (b) $(\forall\, x, y \in A)\ xRy \Rightarrow yRx$; (c) $(\forall x,y,z \in A)\ xRy \wedge yRz \Rightarrow xRz$; (d) all of these; (e) none of these
4. In a reflexive relation on A (a) each element of A is related to itself; (b) each ordered pair (a, b) is matched by (b, a); (c) if aRb and bRc then aRc; (d) the diagonal of the matrix is empty; (e) none of these
5. In a symmetric relation on A (a) each element of A is related to itself; (b) each ordered pair (a, b) is matched by (b, a); (c) if aRb and bRc then aRc; (d) the diagonal of the matrix is empty; (e) none of these
6. In a transitive relation on A (a) each element of A is related to itself; (b) each ordered pair (a, b) is matched by (b, a); (c) if aRb and bRc then aRc; (d) the diagonal of the matrix is empty; (e) none of these
7. If R is an antisymmetric relation over A, and if $(x, y) \in R$, then (a) $x \in A$; (b) $y \notin A$; (c) $(y, x) \notin R$; (d) $x = y$; (e) $x \neq y$
8. Relations that are reflexive, symmetric, and transitive are (a) orderings; (b) partitions; (c) equivalence relations; (d) functions; (e) nonexistent
9. An equivalence relation is induced by (a) inference; (b) quantifiers; (c) commutativity; (d) numeric equality; (e) a partition
10. Equivalence relations are (a) induced by partitions; (b) equal; (c) asymmetric; (d) decidable; (e) intersections

3. Functions

1. A reflexive transitive closure is obtained by (a) applying a function once; (b) applying a function twice; (c) applying a function repeatedly; (d) taking the intersection of two sets; (e) taking the union of two sets
2. If $y = f(x)$ then (a) f is the image of y under x; (b) f is the image of y under x; (c) x is the image of f under y; (d) y is the image of x under f; (e) (c) y is the image of f under x
3. If I_A is the identity function for set A, then $(\forall x \in A)\ I_A(x)$ = (a) 0; (b) 1; (c) x; (d) A; (e) I_A
4. A polynomial is a (a) linear function; (b) exponential function; (c) sum of power functions; (d) numeric value; (e) predicate
5. A bijection is a(n) (a) partition; (b) binary number; (c) one-to-one correspondence; (d) proof; (e) none of these
6. Any bijection has a(n) (a) identity value; (b) inverse function; (c) complement; (d) intersection; (e) transition
7. ____ injections are bijections (a) all; (b) some; (c) no; (d) binary; (e) none of these
8. ____ surjections are bijections (a) all; (b) some; (c) no; (d) binary; (e) none of these
9. ____ bijections are injections (a) all; (b) some; (c) no; (d) binary; (e) none of these
10. ____ bijections are surjections (a) all; (b) some; (c) no; (d) binary; (e) none of these
11. ____ surjections are injections (a) all; (b) some; (c) no; (d) binary; (e) none of these
12. ____ injections are surjections (a) all; (b) some; (c) no; (d) binary; (e) none of these
13. A surjection maps (a) from all elements of its domain; (b) no two values to the same result; (c) randomly; (d) to all elements of its range; (e) none of these
14. A relation in which every left-hand member is paired with not more than one right-hand member is (a) transitive; (b) symmetric; (c) reflexive; (d) a function; (e) none of these

4. Sequences and languages

1. A string is a (a) collection; (b) set; (c) tree; (d) sequence; (e) list
2. A language is a (a) string; (b) number; (c) set of numbers; (d) sequence of strings; (e) set of strings
3. For array A, $|A|$ is (a) the absolute value of the sum of A's elements; (b) the absolute value of A; (c) the smallest element of A; (d) the number of elements in A; (e) none of these
4. An infinite sequence may be defined (a) by enumeration; (b) only by formula for n^{th} term; (c) only recursively; (d) either by formula or recursively; (e) in propositional logic
5. When a function returns $\uparrow$, it (a) returns 0; (b) returns an infinite quantity; (c) is defined; (d) is undefined; (e) is random

6. When a function returns ↓, it (a) returns 0; (b) returns an infinite quantity; (c) is defined; (d) is undefined; (e) is random
7. A sequence over set A is (a) a relation $\subseteq (A \times A)$; (b) a function $f : \mathbf{N} \rightarrow A$; (c) an element of $A \times A$; (d) a language; (e) none of these
8. The sum of elements of a sequence is de4noted using (a) π; (b) Σ; (c) ϕ; (d) δ; (e) Δ
9. Finite sequences may be represented in computer memory using (a) integers; (b) real numbers; (c) arrays; (d) trees; (e) classes
10. In our discussion of languages, λ represents (a) a function; (b) an alphabet; (c) a symbol; (d) a string; (e) none of these
11. In our discussion of languages, Σ is (a) a function; (b) an alphabet; (c) a symbol; (d) a string; (e) none of these
12. An alphabet is a(n) (a) number; (b) string; (c) finite set; (d) symbol; (e) infinite set
13. Σ is by convention (a) finite; (b) countable; (c) uncountable; (d) a sequence; (e) none of these
14. Σ^0 is (a) {λ}; (b) {(0), (1)}; (c) {00, 01, 10, 11}; (d) strings of length k; (e) all strings over Σ
15. Σ^1 is (a) {λ}; (b) {(0), (1)}; (c) {00, 01, 10, 11}; (d) strings of length k; (e) all strings over Σ
16. Σ^2 is (a) {λ}; (b) {(0), (1)}; (c) {00, 01, 10, 11}; (d) strings of length k; (e) all strings over Σ
17. Σ^k is (a) {λ}; (b) {(0), (1)}; (c) {00, 01, 10, 11}; (d) strings of length k; (e) all strings over Σ
18. Σ^* is (a) {λ}; (b) {(0), (1)}; (c) {00, 01, 10, 11}; (d) strings of length k; (e) all strings over Σ
19. Σ^* is (a) a number; (b) a symbol; (c) an alphabet; (d) a language; (e) none of these
20. Concatenation of languages is (a) $L_1 L_2$; (b) L^*; (c) $L_1 - L_2$; (d) $L_1 \cap L_2$; (e) none of these
21. Iteration of language is (a) $L_1 L_2$; (b) L^*; (c) $L_1 - L_2$; (d) $L_1 \cap L_2$; (e) none of these
22. Boolean expressions are defined (a) selectively; (b) iteratively; (c) recursively; (d) transitively; (e) reflexively
23. The language of Boolean expressions is (a) free-form; (b) a set of numbers; (c) a set of recursively-defined strings; (d) the same as regular expressions; (e) a set of proofs in predicate logic
24. An alphabet is (a) finite; (b) infinite; (c) finite or infinite; (d) uncountable; (e) none of these
25. A language is (a) finite; (b) infinite; (c) finite or infinite; (d) uncountable; (e) none of these
26. Regular expressions may be constructed by (a) concatenation, selection, and subtraction; (b) addition and iteration; (c) addition, selection, and iteration; (d) concatenation; (e) concatenation, selection, and iteration

5. Recurrence relations

1. The well-ordering principle asserts that if all elements of a set exceed some value, k, then (a) the set may be arranged in order; (b) a sorting algorithm will work on the set; (c) there exists a minimal element of the set; (d) the set is finite; (e) the value k is in the set
2. The Fibonacci numbers are an instance of a(n) (a) finite set; (b) recursively defined sequence; (c) undecidable set; (d) inductive proof; (e) none of these
3. Peano defined **N** (a) by induction; (b) by contradiction; (c) by enumeration; (d) by encryption; (e) as a subset of **R**
4. Any computable function can be defined (a) by induction; (b) by contradiction; (c) by enumeration; (d) by encryption; (e) as a subset of **R**
5. A recurrence defines (a) a set of natural numbers; (b) a logical formula; (c) a computable function; (d) an undecidable problem; (e) none of these
6. A recursive definition (a) uses a *while* loop; (b) lists all possibilities; (c) uses the term defined; (d) is impossible; (e) is inefficient
7. Recurrences are used in (a) input specification; (b) proofs of correctness; (c) time analysis; (d) type checking; (e) none of these
8. Recurrences (a) are a form of pseudocode; (b) suggest algorithms but not running time; (c) suggest running time but not algorithms; (d) suggest running time and algorithms; (e) none of these
9. Recurrences may help in time analysis if we find (a) count of iterations of *while* loop; (b) clock readings; (c) exit condition; (d) depth of recursion; (e) none of these
10. Recurrence relations enable us to use _____ to obtain running time (a) empirical tests; (b) loop nesting; (c) base-case running time; (d) depth of recursion; (e) base-case running time and depth of recursion
11. The more time-consuming part of the execution of an algorithm defined by a recurrence is (a) the base step; (b) the recursive step; (c) calculation of the time function; (d) proof of correctness; (e) design

6. Big-O, Ω, Θ

1. Vector traversal is O(___) (a) 1; (b) lg n; (c) n; (d) n^2; (e) 2^n
2. A recursive-case running time of $(1 + T(n–1))$ indicates ____ time (a) constant; (b) logarithmic; (c) linear; (d) quadratic; (e) exponential
3. Function g is an upper bound on function f iff for all x, (a) $g(x) \leq f(x)$; (b) $g(x) \geq f(x)$; (c) $g = O(f)$; (d) $f = \Omega(g)$; (e) none of these
4. Function g is a lower bound on function f iff for all x, (a) $g(x) \leq f(x)$; (b) $g(x) \geq f(x)$; (c) $f = O(g)$; (d) $g = \Omega(f)$; (e) none of these
5. Big-Omega notation expresses (a) tight bounds; (b) upper bounds; (c) lower bounds; (d) worst cases; (e) none of these
6. Big-O notation expresses (a) tight bounds; (b) upper bounds; (c) lower bounds; (d) best cases; (e) none of these
7. Theta notation expresses (a) tight bounds; (b) upper bounds; (c) lower bounds; (d) worst cases; (e) none of these
8. $T_\alpha(n) = O(f(n))$ means that (a) algorithm α computes function f; (b) algorithm α produces a result in time at least $f(n)$ for inputs of size n; (c) algorithm α produces a result in time not greater than $f(n)$ for inputs of size n; (d) Algorithm T runs in time α; (e) Algorithm f computes function T on data α
9. $\log_2 n \in O(\text{sqrt}(n))$ means that the logarithm function _____ the square root function (a) grows as fast as; (b) grows no faster than; (c) grows at least as fast as; (d) is in a mapping of real numbers defined by; (e) regardless of parameter produces a result smaller than

10. Quadratic time is faster than (a) O(1); (b) O(lg n); (c) O(n^2); (d) O(n^3); (e) none of these
11. The theorem, $T_1(n) \in O(g_1(n)) \wedge T_2(n) \in O(g_2(n)) \Rightarrow T_1(n) + T_2(n) \in O(\max\{g_1(n), g_2(n)\})$ says that (a) the slower and faster parts of an algorithm together set its running time; (b) the faster part of an algorithm dominates in determining running time; (c) the slower part of an algorithm dominates in determining running time; (d) Algorithm T computes functions g_1 and g_2; (e) Algorithm T finds the maximum of g_1 and g_2
12. When the running time for the base case of a recursive algorithm is O(n) and the remaining part of input to process is reduced by one at each recursive step, the total running time is (a) O(1); (b) O(lg n); (c) O(n lg n); (d) O(n); (e) O(n^2)
13. In a recursive algorithm, when the running time for the base case is O(1) and remaining work of an algorithm is reduced by one at each step, the running time is (a) O(1); (b) O(lg n); (c) O(n lg n); (d) O(n); (e) O(n^2)
14. A recursive-case running time of ($n + T(n–1)$) indicates ____ time (a) constant; (b) logarithmic; (c) linear; (d) quadratic; (e) exponential

Terminology for topic 2 (Sets, relations, recurrences)

antisymmetric relation
arity
array
associative property
big-O notation
big-omega notation
bijection
Cartesian product
co-domain
complement
contradiction
disjoint sets
distributive property
equivalence class
equivalence relation
idempotent property
identity function
image of *x* under *f*
index
injection
inverse
language
linear function
partial order
partition
Peano's axioms
power function
proper subset
range
recurrence relation
reflexive relation
reflexive transitive closure
relation
sequence
surjection
symmetric relation
theta notation
transitive relation
universal set
upper bound

Problems to assess outcomes for topic 2

2.1a Explain or apply a concept in set theory (essential)

1. What is the *universal set*?
2. What is the *complement* of a set?

Explain the value and meaning of

3. $(B \subseteq A)$
4. $(B \supset A)$
5. $A \cap B$
6. $A \cup B$
7. $A - B$
8. $A \cap \varnothing$
9. $A \cup \varnothing$

Let $A = \{1, 3, 5, 7\}$, $B = \{3, 4, 5\}$. *Enumerate:*

10. $A \cap B$ {3,5}
11. $A \cup B$ {1,3,4,5,7}
12. $A - B$ {1, 7}

2.1b Prove a theorem in set theory (essential)

Prove that for all sets *A*, *B* and *C*,

1. $(B - A) \subseteq B \cap A^c$
2. $(A \subseteq C \ \cap B \subseteq C) \Rightarrow (A \cup B) \subseteq C$
3. $(A - B) \cup (C - B) = (A \cup C) - B$
4. $(A \subseteq B) \Rightarrow (A \cap C) \subseteq (B \cap C)$
5. $A \cap \varnothing = \varnothing$
6. $A \cup \varnothing = A$
7. if $A \subseteq B$ then $B^c \subseteq A^c$

2.2a Describe a relation (essential)

1. Describe a *relation* between {1, 2} and {*a*, *b*, *c*}.
2. What is meant by a *relation* between sets *A* and *B*?
3. What is meant by a *relation on* set $S = \{a, b, c, d\}$?
4. Enumerate the *greater-than relation* on {1, 2, 3}.
5. What is the *reflexive transitive closure* of a relation?
6. For sets *A*, *B*, describe $(A \times B)$.
7. What is the largest relation on set *A*?

2.2b Apply the notion of an equivalence relation*

(1-4) Is the relation below an equivalence relation? Justify the three parts of your answer.

1. {(1, 2), (2, 1), (1, 3), (3, 1)}
2. {(0, 0), (0, 1), (0, 2), (1, 2), (2,0)}
3. {(1, 2), (2, 1), (2, 3), (3, 2)}
4. {(1, 1), (2, 2), (3, 3), (1, 2), (2,1)}

5. What is a *reflexive* relation?
6. What is a *symmetric* relation?
7. What is a *transive* relation?
8. What relations are *equivalence* relations?
9. Describe and name the set of relations that *partition* sets.

2.3a Describe a function (essential)

1. Distinguish *relations* from *functions*.
2. What are the *polynomial* functions?
3. What are the *exponential* functions?
4. Distinguish *partial* from *total* functions.
5. Distinguish *sets* from *functions*.
6. Distinguish the *domain* of a function from its *range*.
7. What is the relationship of $f : A \rightarrow B$ to $A \times B$?
8. What are the domain and the range of the square-root function?
9. Explain how the arithmetic operators are functions – of what arity?
10. Distinguish *predicates* from *functions*.
11. Identify and give an example of $f : \mathrm{N} \rightarrow \{\mathrm{F}, \mathrm{T}\}$

Is the following a function? If not, why not?

12. {(1, 2), (2, 1), (2, 3), (3, 1)}
13. {(0, 0), (1, 1), (2, 2), (2, 3)}

2.3b Define a class of functions

1. What is the *inverse* of an exponential function?
2. What is the *identity function*?
3. What is the *inverse* of the square-root function?
4. For $f(x)$, what is the *inverse* of *f*, and what property does it have with respect to $f(x)$?
5. Distinguish *injections* from *surjections*.
6. What is a *bijection*?

2.4a Use a function to define a sequence (essential)

1. Explain how a sequence is a function.

Write a definition of the function that specifies the following sequence:

2. the powers of 2
3. the numbers that are each the sums of the linear series from 1 to *n*
4. the squares of natural numbers
5. the numbers that are each the product of all the whole numbers from 1 to *n*

2.4b Define a language (essential)

Using a regular expression, define the language of strings over {0, 1} in which

1. the second symbol is a 1.
2. two consecutive 0s occur.
3. an even number of 1's occur.
4. the last symbol is 0.
5. no two consecutive symbols are the same.

2.5a Describe a recursively defined function (essential)

1. Describe the *factorial* function. What is recursive about it?
2. What is the name of a particular mathematical technique or notation, for defining a function, that converts straightforwardly into code or pseudocode?

What are the following? What is recursive about them?

3. $f(a,b) = \begin{cases} 0 & \text{if } a = 0 \\ b + f(a-1,b) & \text{otherwise} \end{cases}$
4. $h(a,b) = \begin{cases} 0 & \text{if } a = 0 \\ b \times h(a-1,b) & \text{otherwise} \end{cases}$
5. $j(a) = \begin{cases} 0 & \text{if } a = 0 \\ 2 + j(a-1) & \text{otherwise} \end{cases}$
6. $g(a,b) = \begin{cases} 0 & \text{if } a = 0 \\ b + g(\lfloor a \div 2 \rfloor, 2b) & \text{if } a \text{ is odd} \\ g(\lfloor a \div 2 \rfloor, 2b) & \text{if } a \text{ is even} \end{cases}$

2.5b Write a recurrence to define a function

Use a recurrences to define the following functions:.

1. **Sum (*A*)**
   > Computes sum of
   > array elements
   y ← 0
   i ← 1
   while *i* ≤ |*A*|
   y ← *y* + *A*[*i*]
   i ← *i* + 1
   return *y*

2. **Product (x, y)**
   >Performs multiplication
   result ← 0
   For *i* ← 1 to *x*
   result ← *result* + *y*
   Return *result*

3. **Max (*A*)**
   > Returns largest elt of *A*
   y ← *A*[1]
   i ← 1
   while *i* < |*A*|
   if *y* < *A*[*i*]
   y ← *A*[*i*]
   i ← *i* + *1*
   return *y*

4. **Pow (a, b)**
   > returns a^b
   y ← *a*
   i ← 1
   while i < b
   y ← *a* × *y*
   i ← *i* + 1
   return *y*

5. **Fact (*x*)**
   > Computes factorial:
   y ← 1
   i ← 1
   while i < *x*
   y ← *i* × *y*
   i ← *i* + 1
   return *y*

2.6a Define O, Ω, and Θ notation

1. What is the main notation for expressing the complexity of algorithms as *tight bounds*? How does it compare with the other commonly used notations?
2. For function *f*, define and describe O(*f*).
3. What is the main notation for expressing the complexity of algorithms as *upper bounds*? How does it compare with the other commonly used notations?
4. For function *f*, define and describe θ(*f*).
5. What is the main notation for expressing the complexity of algorithms as *lower bounds?* How does it compare with the other commonly used notations?
6. For function *f*, define and describe Ω(*f*).
7. Use two simple formulas to show the relation among O, Θ, and Ω notations. (*Hint:* if $f(n) \in \Omega(g(n))$, what follows?)

Multiple-choice questions on Topic 3 (Graphs)

1. Graphs

1. A graph is (a) a set of integers; (b) a set of vertices; (c) a set of vertices and a set of edges; (d) a set of edges; (e) a set of paths
2. The degree of a vertex in a graph is (a) the number of vertices in its graph; (b) the number of edges in its graph; (c) the number of paths; (d) the number of distinct connected subgraphs; (e) the number of other vertices adjacent to it
3. A graph is defined in part by (a) exactly one ordered pair of vertices; (b) a relation; (c) a cycle; (d) one path joining each pair of vertices; (e) none of these.
4. A series of edges that connect two vertices is called (a) a path; (b) a cycle; (c) a connection; (d) a tree; (e) a collection
5. To design a communications network that joins all nodes without excessive lines, we must find a (a) path; (b) connectivity number; (c) minimal spanning three; (d) expression tree; (e) search tree
6. A repeating series of edges that form a path from a vertex to itself is (a) a spanning path; (b) a cycle; (c) a connection; (d) a tree; (e) an edge
7. A weighted graph has an adjacency matrix that is (a) integers; (b) vertices; (c) real numbers and ∞; (d) booleans; (e) none of these
8. The prerequesite relationships among required courses in the Computer Science major form a (a) binary tree; (b) linked list; (c) directed acyclic graph; (d) weighted graph; (e) spanning tree
9. A tree is a graph that is (a) connected and cyclic; (b) connected and acyclic; (c) unconnected and cyclic; (d) unconnected and acyclic; (e) none of these
10. A graph may be fully represented by (a) its vertices; (b) its edges; (c) an adjacency matrix; (d) the degrees of its vertices; (e) none of these
11. The breadth-first search (a) uses a queue; (b) uses a stack; (c) searches an array; (d) searches a tree; (e) none of these
12. The depth-first search (a) uses a queue; (b) uses a stack; (c) searches an array; (d) searches a tree; (e) none of these

2. Graph isomorphism

1. Graph path search involves finding a (a) set of vertices; (b) sequence of vertices; (c) set of edges; (d) minimal set of edges; (e) none of these
2. Two graphs are isomorphic iff (a) they have the same numbers of vertices and edges; (b) they have the same degrees; (c) bijections of a special kind exist between their sets of vertices and edges; (d) they have no vertices in common; (e) one is a subgraph of the other
3. Graphs for which bijections of a special kind exist between their sets of vertices and edges are (a) nested; (b) transitive; (c) undecidable; (d) disjoint; (e) isomorphic
4. Graphs that have the same structure are (a) nested; (b) transitive; (c) undecidable; (d) disjoint; (e) isomorphic
5. Graph isomorphism invariant properties include (a) having the same numbers of vertices and edges; (b) satisfiability; (c) reachability; (d) well ordering; (e) well foundedness

3. Transition systems

1. A transition system is defined by (a) a set of states and a relation on them; (b) a set of points and a mapping among them; (c) a set of symbols and rules for sequencing them; (d) a set of strings; (e) none of these
2. A transition system is (a) an interactive system; (b) a labeled graph denoting states and transitions; (c) an algorithm; (d) a set of equations; (e) a language
3. A state-transition system with probabilistic transitions is a(n) (a) semantic net; (b) Bayesian net; (c) finite automaton; (d) Turing machine; (e) Markov chain
4. Transitions that are probability functions of a current state characterize (a) finite automata; (b) Bayesian networks; (c) schemas; (d) Markov models; (e) none of these
5. In our discussion of DFAs, δ is (a) a function; (b) an alphabet; (c) a symbol; (d) a string; (e) none of these
6. The reflexive transitive closure of δ maps from (a) states to states; (b) states and symbols to states; (c) states and strings to states; (d) states and symbols to symbols; (e) none of these
7. Whether a certain string belongs to the language recognized by a finite automaton is determined by (a) the output; (b) the transition; (c) whether the automaton terminates; (d) whether the automaton terminates in an accepting state; (e) none of these
8. For each finite automaton there exist(s) ___ corresponding language(s) (a) no; (b) one; (c) two; (d) some finite number of; (e) infinitely many
9. The Turing machine model is said to capture (a) regular languages; (b) interaction; (c) efficient computation; (d) algorithmic computation; (e) all of these
10. A Turing machine has ____ storage (a) random-access; (b) limited; (c) unbounded; (d) stack; (e) queue
11. A Turing machine (a) lacks an alphabet; (b) has tape instead of states; (c) can compute any mathematical function; (d) stores data on a tape; (e) none of these

4. Structural induction

1. Structural induction may be used to show properties of (a) sets of integers; (b) real numbers; (c) sets of strings; (d) algorithms; (e) none of these
2. We may use _____ to prove that all elements of a certain language have equal numbers of left and right parentheses (a) contradiction; (b) enumeration; (c) counter example; (d) structural induction; (e) strong induction

Terminology for topic 3 (Graphs and transition systems)

acyclic graph	digraph	Markov assumption	pushdown automata	subgraph
adjacency matrix	edge	Markov decision process	reactive system	temporal logic
computation tree	finite automaton	Markov model	reflexive transitive	transition function
Computation Tree Logic	finite transducer	matrix	closure	transition system
connected graph	graph	minimal spanning tree	regular expression	Turing machine
cycle	isomorphism	model checking	regular language	weighted graph
degree	Kripke structure	path	structural induction	

Problems to assess outcomes for topic 3

3.1a Construct a graph from a description (essential)

Draw a graph of the following:

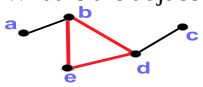
1. {(1, 2), (2, 1), (1, 3), (3, 1)}
2. {(0, 0), (0, 1), (0, 2), (1, 2), (2,0)}
3. {(1, 2), (2, 1), (2, 3), (3, 2)}
4. {(1, 1), (2, 2), (3, 3), (1, 2), (2,1)}

Draw a graph with these properties:

5. Five vertices of degrees 1, 3, 3, 1, 2
6. Five vertices of degrees 1, 2, 3, 2, 2
7. Six vertices of degrees 2, 2, 3, 3, 2, 2
8. Draw the digraph with vertices {*a*, *b*, *c*} and with the following adjacency matrix:

0	1	1
2	0	1
0	2	1

9. What is the adjacency matrix of the following graph?



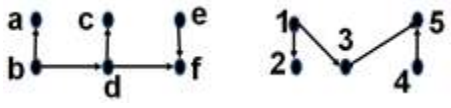
3.1b Describe a basic concept of graph theory (essential)

1. What is a *path*? Give a special classes of paths.
2. What is a *cycle*?
3. What is the *degree* of a vertex; of a graph?
4. When is $G' = (V', E')$ a *subgraph* of $G = (V, E)$?
5. Describe the *adjacency matrix* of a *weighted graph*.

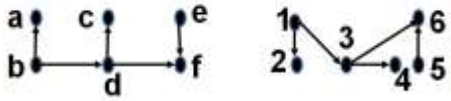
3.2 Apply the concept of graph isomorphism

If two graphs side by side below are isomorphic, then give the two functions that define an isomorphism. Otherwise, give an isomorphism invariant not shared by them.

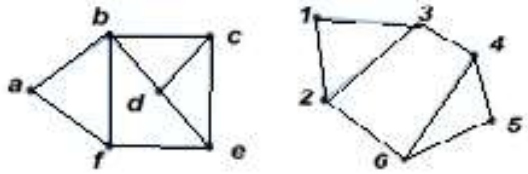
1.



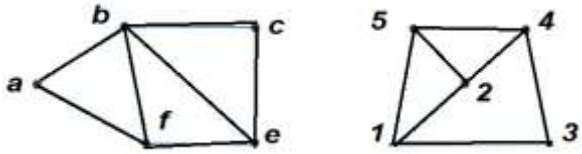
2.



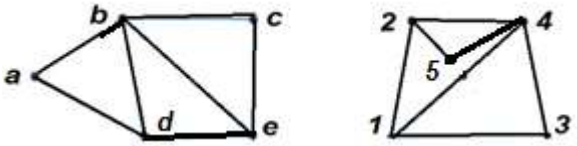
3.



4.

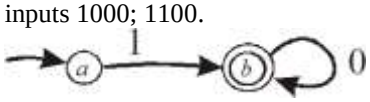


5.

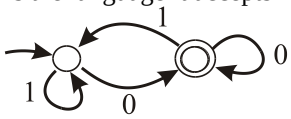


3.3 Describe a transition system (priority)

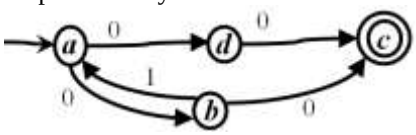
1. Describe the components and execution of a *transition system*.
2. Describe the steps taken by the transition system below on inputs 1000; 1100.



3. How many states does the transition system below have? Is the language it accepts finite or infinite? Why?



4. What are the *states* of the system below? In what way are certain ones different from the others in a way that affects the output of the system?



5. Give the *transition function* of the transition system above.

3.4 Use structural induction to prove an assertion about graphs (priority)

Let graph $G = (V, E)$, where $|G|$ is the number of vertices and edges; $|V|$ is the number of vertices; $|E|$ is the number of edges. Prove *by structural induction*:

1. For any graph, the sum of the degrees of the vertices is even.
2. For any graph, the number of vertices with odd degree is even.
3. If G is a connected graph with n vertices and $(n – 1)$ edges, then G is acyclic.
4. Removing an edge from a acyclic graph yields a graph that is not connected.
5. The sum of the degrees of all vertices in a graph is twice the number of edges.
6. The result of removing an edge from an acyclic graph adds one to its connectivity number.

Multiple-choice questions on Topic 4 (Trees)

1. Properties of trees

1. To model a hierarchy, it is most convenient to use a(n) (a) simple type; (b) array; (c) linked list; (d) binary search tree; (e) general tree
2. A tree has no (a) edges; (b) vertices; (c) paths; (d) cycles; (e) connectivity
3. A node in a tree that is the child of no other node is called the (a) leaf; (b) parent; (c) root; (d) ancestor; (e) none of these
4. A leaf node is one without (a) data; (b) children; (c) a parent; (d) references pointing to it; (e) none of these
5. The maximum path length from the root to a leaf is a tree's (a) degree; (b) connectivity number; (c) depth; (d) edge count; (e) vertex count
6. In a tree, any two vertices are connected by ____ distinct path or paths. (a) no; (b) exactly one; (c) one or more; (d) many; (e) exactly two
7. A root vertex in a tree may have (a) a parent; (b) siblings; (c) children; (d) decidability; (e) mentors
8. A binary tree has (a) one branch; (b) two vertices; (c) two paths; (d) exactly two edges from each vertex; (e) up to two edges from each vertex
9. A structure that is connected and contains all the vertices in a weighted graph is (a) a coloring; (b) a path; (c) a spanning tree; (d) a single-source shortest path; (e) a depth-first traversal
10. A minimal spanning tree can be found by (a) subtracting edges greedily; (b) adding edges greedily; (c) seeking the shortest path; (d) recursive traversal; (e) none of these

2. Using tree efficiency

1. The depth of the decision tree for an algorithm expresses its (a) correctness; (b) problem class; (c) running time; (d) space requirement; (e) data arrangement
2. A full binary tree with k leaves has height (a) k; (b) 2^k; (c) $2k$; (d) $\log_2 k$; (e) none of these
3. A logarithmic function is the inverse of an ___ function (a) addition; (b) exponential; (c) reciprocal; (d) multiplication; (e) factorial
4. The inverse of an exponential function is a (a) difference; (b) reciprocal; (c) division; (d) logarithm; (e) power
5. The depth of a heap of size n is close to (a) 1; (b) $\log_2 n$; (c) the square root of n; (d) $n / 2$; (e) n^2
6. After each step of the BST search, the quantity of remaining data to be searched is on average (a) 1; (b) lg n; (c) $n \div 2$; (d) n; (e) $2n$
7. The height of a BST is on average O(__) (a) 1; (b) lg n; (c) n; (d) n lg n; (e) n^2
8. After each step of the binary search, the quantity of remaining data to be searched is on average (a) 1; (b) lg n; (c) $n \div 2$; (d) n; (e) $2n$
9. A structure that shows possible outcomes of all steps of a computation is a (a) flowchart; (b) module hierarchy; (c) binary tree; (d) decision tree; (e) none of these
10. When the quantity of remaining data to be processed in an algorithm, at each step, is $(n \div 2)$, the complexity is O(___) (a) 1; (b) lg n; (c) n; (d) n^2; (e) 2^n
11. A recursive-case running time of $(1 + T(n \div 2))$ indicates ____ time (a) constant; (b) logarithmic; (c) linear; (d) quadratic; (e) exponential
12. A recursive-case running time of $(n + T(n \div 2))$ indicates ____ time (a) constant; (b) n lg n; (c) linear; (d) quadratic; (e) exponential
13. When base case is O(n) and remaining work of an algorithm is cut in half at each step, the running time is (a) O(1); (b) O(lg n); (c) O(n lg n); (d) O(n); (e) O(n^2)
14. Decrease by constant factor is consistent with $T(n)$ = (a) $T(n-1) + O(1)$; (b) $T(n-1) + O(n)$; (c) $T(n / b) + O(1)$; (d) $2T(n-1) + O(1)$; (e) none of these
15. $T(n) = T(n / b) + f(n)$ is consistent with (a) decrease by constant; (b) decrease by one; (c) decrease by constant factor; (d) O(n^2); (e) O(n)
16. When base case is O(1) and remaining work of an algorithm is cut in half at each step, the running time is (a) O(1); (b) O(lg n); (c) O(n lg n); (d) O(n); (e) O(n^2)
17. The Master Theorem (Main Recurrence Theorem) (a) is used to prove correctness of algorithms; (b) gives general solutions to time recurrences for divide-and-conquer algorithms; (c) gives intractability results; (d) is proven by temporal logic; (e) none of these
18. Binary search can be shown to be $\Theta(\lg n)$ by (a) Hoare triples; (b) temporal logic; (c) predicate logic; (d) the Master Theorem (Main Recurrence Theorem); (e) induction

3. Applications of trees in AI and bioinformatics

1. A state space is a set of (a) paths; (b) locations in the physical universe; (c) governmental entities; (d) actual arrangements of values; (e) possible arrangements of values
2. Games and puzzles are simple examples of (a) embodied intelligence; (b) state-space search; (c) inference; (d) agent interaction; (e) adaptation
3. A set of possible arrangements of values is a(n) (a) state space; (b) path; (c) combination; (d) random variable; (e) none of these
4. A state space is (a) part of RAM; (b) one set of variable assignments; (c) a set of possible arrangements of values; (d) a graph; (e) none of these
5. In a game tree, vertices are (a) cities; (b) players; (c) moves; (d) board positions; (e) pieces
6. A phylogenetic tree represents (a) an algorithm; (b) a data structure; (c) a taxonomy; (d) a game; (e) a problem
7. Applications in bioinformatics make use of ___ trees to determine ancestry (a) evergreen; (b) decision; (c) phylogenetic; (d) binary-search; (e) complete binary

Terminology for topic 4 (Trees)

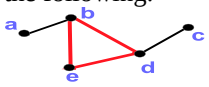
binary search	complete binary tree	heap	logarithmic function	rooted tree
binary search tree	decision trees	height	*m*-ary tree	state-space search
binary tree	exponential function	internal vertex	Master Theorem	subtree
BST	full binary tree	leaf	parent	tree
child	game trees	level	root	

Questions to assess outcomes for topic 4

4.1a Draw a tree with given specifications (priority)

Draw a tree with

1. six vertices, at least one of which is neither a root nor a leaf.
2. six vertices, one of which is of degree 3.
3. five vertices, at least one of which is both a root and a leaf.
4. six vertices, four of which are leaf nodes.
5. seven vertices that is a complete binary tree
6. five vertices, two of which are leaves
7. three leaves, that is a subgraph of the following:



4.1b Describe and prove a property of trees (priority)

Prove by induction:

1. If a graph $G = (V, E)$ is connected, and $|V| = |E| + 1$, then G is a tree.
2. Any graph has a subgraph that is a tree.
3. Any tree with more than one vertex has a vertex of degree one.
4. Exactly one path joins any pair of vertices in a tree.
5. Removing an edge from a tree disconnects it.
6. Adding an edge to a tree creates a cycle.
7. For any full *m*-ary tree (tree in which every non-leaf node has exactly *m* children) T, $| T |$ mod $m = 1$.
8. For any tree $T = (V, E)$, $| V | = | E | + 1$.
9. There is exactly one path between any two vertices in a tree.
10. The height of a complete binary tree with n vertices is $\log_2 n$.
11. Every full binary tree has $2^k - 1$ vertices, where k is the depth of the tree.
12. Any tree with more than one vertex has more than one vertex of degree 1.

4.2a Explain the running time of a tree-enabled algorithm (priority)

In terms of the number of vertices, explain the running time of

1. BST search (average case)
2. traversing a path of a *heap* from the root to a leaf.
3. BST insertion (average case)
4. deleting all items from a balanced BST of height n
5. descending from the root to a leaf of a balanced tree with 2^n nodes
6. performing addition on pairs of binary numerals that range in value from 0 to n
7. the binary-search algorithm

(8-9) Give the complexities of the algorithms below and justify your answers.

8. ***Alg-1* (*root, key*)**

```
If root = null
    return false
If data (root) = key
    return true
otherwise
    if data(root) > key
        return Alg-1 (left (root), key))
    otherwise
        return Alg-1 (right (root), key))
```

9. **Alg-2 (*A, first, last*)**

```
if first > last   // (i.e., nothing to search)
    return false
else
    middle ← (first + last) ÷ 2
if A[middle] matches key
    return true
otherwise
    if A[middle] > key
        return Alg-4(A, first, middle - 1, key)
    else
        return Alg-4 (A, middle + 1, last, key)
```

4.2b Apply the Master Theorem to solve a recurrence

Use the Master Theorem[1] to derive a tight-bound (Θ) solution to the following recurrences. Show your work.

1. $T(n) = 3T(n / 2) + \Theta(n)$
2. $T(n) = 2T(n / 2) + \Theta(1)$
3. $T(n) = 4T(n / 3) + \Theta(n^2)$
4. $T(n) = T(n) + \Theta(\lg n)$
5. $T(n) = 3T(n / 2) + \Theta(n^2)$
6. $T(n) = 2T(n / 2) + \Theta(n \lg n)$
7. $T(n) = 3T(n / 3) + \Theta(1)$
8. $T(n) = 3T(n / 5) + \Theta(n^3)$
9. $T(n) = 4T(n / 2) + \Theta(1/n)$
10. $T(n) = 3T(n / 4) + \Theta(1)$

4.3 Describe an AI or bioinformatics application of trees

(1-3) Describe the following and how they are used.

1. game trees
2. phylogenetic trees
3. decision trees

4. What does a tree represent in state-space search? Describe its role.
5. Describe how trees may be used in bioinformatics, with specific reference to tree structure.
6. Describe the tree structure of the state space search in tic tac toe.
7. Describe the tree structure of the state space to search in the game of chess.

[1] Let $T(n) = aT(\lfloor n/b \rfloor) + f(n)$, with $f(n) \in \theta(n^d)$, $d \geq 0$. Then $T(n) \in \theta(n^d)$, if $a < b^d$; $\theta(n^d \lg n)$, if $a = b^d$; $\theta(n \log_b a)$, if $a > b^d$

Multiple-choice questions on Topic 5 (Countability)

1. Countable sets

1. An alphabet is (a) finite; (b) infinite; (c) finite or infinite; (d) uncountable; (e) none of these
2. The set of strings of length *k*, over a finite alphabet, for given constant *k*, is (a) countably infinite; (b) finite; (c) uncountable; (d) undecidable; (e) none of these
3. Two sets have the same cardinality if (a) they are both finite; (b) neither is strictly included in the other; (c) a bijection exists between them; (d) they are both infinite; (e) none of these
4. The natural numbers (a) can be paired up with the reals; (b) are countable; (c) are uncountable; (d) are as numerous as any set; (e) none of these
5. Σ^* is (a) finite; (b) countable; (c) uncountable; (d) an alphabet; (e) none of these
6. A language is (a) finite; (b) infinite; (c) finite or infinite; (d) uncountable; (e) none of these
7. Two sets have the same cardinality iff there is a ____ between them (a) relation; (b) bijection; (c) function; (d) assertion; (e) injection
8. The cardinality of the set of *natural* numbers is ____ the cardinality of the set of *rational* numbers (a) the same as; (b) greater than; (c) less than; (d) not comparable to; (e) none of these
9. The set of natural numbers is (a) indescribable; (b) finite; (c) countably infinite; (d) uncountably infinite; (e) none of these
10. The set of Java programs is (a) small; (b) finite in number; (c) countable; (d) uncountable; (e) tested
11. Enumerability is an attribute of ____ sets (a) no; (b) all; (c) all infinite; (d) all countable; (e) none of these
12. Which of these sets is countable? i. strings ii. streams iii natural numbers iv real numbers. (a) i and ii; (b) i and iii; (c) ii and iii; (d) ii and iv; (e) none of these
13. Cantor showed that the reals are (a) infinite; (b) countable; (c) uncountable; (d) dense; (e) none of these
14. Cantor's proof about the cardinalities of real and natural numbers was by (a) induction; (b) diagonalization; (c) construction; (d) statistical methods; (e) none of these
15. What proof method was used by Cantor to show that the reals are uncountable? (a) inductive; (b) diagonal; (c) constructive; (d) immediate; (e) none of these
16. A diagonal proof is by (a) induction; (b) contradiction; (c) construction; (d) statistical methods; (e) none of these
17. By what proof method was it shown that the real numbers are uncountable? (a) direct; (b) induction; (c) diagonal; (d) counter-example; (e) none of these
18. The cardinality of the set of real numbers is ____ the cardinality of the set of *rational* numbers (a) the same as; (b) greater than; (c) less than; (d) not comparable to; (e) none of these
19. The set of real numbers is (a) indescribable; (b) finite; (c) countably infinite; (d) uncountably infinite; (e) none of these
20. We can disprove the existence of an enumeration of all the real numbers by assuming an enumeration exists and defining real whose *n*th digit, for all *n*, is different from ____ digit of the *n* the real in the supposed enumeration (a) each; (b) the first; (c) the *n*th; (d) the (*n*+1)th; (e) the last
21. The number of predicates on a set of cardinality *n* is (a) *n*; (b) 2*n*; (c) n^2; (d) 2^n; (e) none
22. The predicates on natural numbers are (a) few; (b) finite in number; (c) countable; (d) uncountable; (e) none

2. Incompleteness

1. Soundness is (a) completeness; (b) validity; (c) consistency; (d) truth; (e) provability
2. A logical system in which no false assertion can be proven is (a) consistent; (b) complete; (c) ambiguous; (d) paradoxical; (e) none of these
3. Godel showed that every consistent system is (a) true; (b) unsound; (c) incomplete; (d) ambiguous; (e) sound
4. A logical system in which every true assertion can be proven is (a) consistent; (b) complete; (c) ambiguous; (d) paradoxical; (e) none of these
5. Gödel numbers (a) are cardinalities; (b) are reals; (c) encode assertions; (d) encode programs; (e) none of these
6. Gödel's incompleteness theorem was proven by (a) induction; (b) diagonalization; (c) construction; (d) statistical methods; (e) none of these
7. A logical system is *complete* iff (a) every assertion is true; (b) every assertion is provable; (c) every true assertion is provable; (d) no false assertion is provable; (e) a theorem exists for every proof
8. A logical system is *consistent* iff (a) every assertion is true; (b) every assertion is provable; (c) every true assertion is provable; (d) no false assertion is provable; (e) a theorem exists for every proof
9. Completeness is ___ soundness (a) equivalent to; (b) stronger than; (c) weaker than; (d) incompatible with; (e) dependent on
10. A system in which every true assertion is provable is (a) satisfiable; (b) valid; (c) complete; (d) consistent; (e) sound

3. Recursive functions

1. Algorithmically computable functions are the same as (a) those computable on a DFA; (b) those computable on a PDA; (c) μ-recursive functions; (d) control devices; (e) none of these
2. The *basic* primitive recursive functions include (a) successor; (b) addition; (c) multiplication; (d) composition; (e) recursion
3. The ___ function is *not* a basic primitive recursive function (a) factorial; (b) zero; (c) successor; (d) projection; (e) predecessor
4. One computable operation on primitive recursive functions is (a) inverse; (b) search; (c) composition; (d) integration; (e) none of these

5. Obtaining a function by primitive recursion is a way to show that the function is (a) continuous; (b) a predicate; (c) computable; (d) uncomputable; (e) none of these
6. The result of minimalization of a primitive recursive function is (a) primitive recursive; (b) computable; (c) minimal; (d) undecidable; (e) none of these
7. All computable functions $f : \Sigma^* \times \Sigma^*$ are (a) primitive recursive; (b) μ-recursive; (c) compositions; (d) undefined; (e) predicates
8. The composition of two computable functions is (a) computable; (b) undefined; (c) time consuming; (d) uncomputable; (e) uncountable
9. Primitive recursion is a way to implement (a) interaction; (b) negation; (c) loops; (d) branches; (e) infinite sets

4. Undecidable problems

1. $f(x)\downarrow$ means (a) $f(x)$ is descending as x rises; (b) $f(x)$ is unknown; (c) f is defined for parameter x; (d) f is undefined for parameter x; (e) none of these
2. $f(x)\uparrow$ means (a) $f(x)$ is descending as x rises; (b) $f(x)$ is unknown; (c) f is defined for parameter x; (d) f is undefined for parameter x; (e) none of these
3. Any computable function can be computed by some (a) DFA; (b) NFA; (c) PDA; (d) Turing machine; (e) none of these
4. Any computable function can be computed by some (a) DFA; (b) NFA; (c) PDA; (d) Java program; (e) none of these
5. Decision problems are equivalent to functions that return (a) natural numbers; (b) strings; (c) truth values; (d) Turing machines; (e) none of these
6. The Halting Problem involves (a) testing a Turing machine to see if it halts; (b) determining from the description of a TM whether it halts; (c) determining how to change the transition function of a TM to cause it to halt; (d) determining what a TM outputs; (e) causing a TM to halt
7. The standard proof that the Halting Problem is undecidable is by (a) induction; (b) indirection; (c) contradiction; (d) indirection; (e) none of these
8. The Halting Problem (a) is decidable; (b) provides an example of a language that no TM accepts; (c) is exponential-time; (d) is a machine; (e) none of these
9. Uncomputable functions correspond to problems that are called (a) undecidable; (b) intractable; (c) P-time; (d) optimization; (e) none of these
10. The Halting Problem is (a) undecidable; (b) intractable; (c) NP-complete; (d) optimization; (e) none of these
11. Decision problems can also be considered as (a) formulas in propositional logic; (b) assertions; (c) array manipulations; (d) languages; (e) none of these
12. ϕ_P is (a) a problem; (b) an algorithm; (c) the function computed by program P; (d) the time function of program P; (e) none of these

5. Non-well-founded sets and coinduction

1. Induction is used to define (a) branch control structures; (b) finite objects; (c) infinite objects; (d) finite sets; (e) none of these
2. Coinduction is used to define sets of (a) branch control structures; (b) finite objects; (c) infinite objects; (d) finite sets; (e) none of these
3. A coinductive definition has no (a) base case; (b) inductive case; (c) endpoint; (d) purpose; (e) none of these
4. Coinduction may define sets of (a) numbers; (b) programs; (c) proofs; (d) strings; (e) streams
5. The wellfoundedness axiom states that the notion of a set belonging to itself is (a) well-founded; (b) meaningless; (c) doubtful; (d) mandatory; (e) none of these
6. Σ^∞ is a set of (a) numbers; (b) symbols; (c) strings; (d) streams; (e) none of these
7. Σ^∞ is (a) finite; (b) countable; (c) uncountable; (d) an alphabet; (e) none of these

Terminology for topic 5 (Countability and computability)

Anti-Foundation Axiom
bijection
bounded minimalization
cardinality
coinduction
completeness
composition of functions
computable
countable set
diagonal proof
finite set
Foundation Axiom
Gödel's theorem
HALT problem
incompleteness
infinite set
injection
minimalization
μ-recursive function
non-well-founded set
one-to-one correspondence
one-to-one function
onto function
primitive recursive function
recursion theory
recursively definable
soundness
stream
surjection
uncomputable function
uncountable set
undecidable problem

Problems to assess outcomes for topic 5

5.1a Prove that a set is countable (priority)

Show that the following sets are countable:

1. binary numerals
2. pairs of natural numbers
3. natural numbers that are multiples of 5
4. English-language sentences
5. Java programs
6. formulas in predicate logic
7. proofs in predicate logic
8. rational numbers
9. finite bit vectors
10. prime numbers
11. even numbers
12. numbers with an even number of digits
13. squares of natural numbers

5.1b Prove that a set is uncountable (priority)

For #1-5, use the diagonal method to show uncountability of the following sets:

1. predicates over natural numbers
2. real numbers
3. predicates over strings
4. the *set of sets* of natural numbers
5. languages over an alphabet
6. *infinite sequences* of ordered pairs of natural numbers
7. Explain what this diagram is used to show.

	1	2	3	4	5	6
1	$b_{1,1}$	$b_{1,2}$	$b_{1,3}$	$b_{1,4}$	$b_{1,5}$	$b_{1,6}$
2	$b_{2,1}$	$b_{2,2}$	$b_{2,3}$	$b_{2,4}$	$b_{2,5}$	$b_{2,6}$
3	$b_{3,1}$	$b_{3,2}$	$b_{3,3}$	$b_{3,4}$	$b_{3,5}$	$b_{3,6}$
4	$b_{4,1}$	$b_{4,2}$	$b_{4,3}$	$b_{4,4}$	$b_{4,5}$	$b_{4,6}$

8. Consider the set of infinite streams of ASCII characters.
   (a) Show that it is uncountable.
   (b) Name the proof method
   (c) For each and every sequence in this set, does there exist a Java program with no input, but with an infinite output loop, that outputs the sequence? Why or why not?
9. Consider a mobile robot that at each step of its existence, must decide whether to turn left or right 5 degrees, or go forward, based on its percept and state at that instant. Define a robot's *output behavior* as a set of infinite sequences of outputs in the set {*left, forward, right*}. Use Cantor's diagonal proof method to show that the set of all possible robot behaviors is uncountable.
10. Answer the following "refutation" of Cantor's proof: "Look, you show me a particular ordering of strings and prove that *this* enumeration omits some real number. So *one* way to list all real numbers fails by being incomplete. So what? Maybe someone could come up with a different ordering that would include all reals."

5.2 Describe the Incompleteness Theorem

1. What is *incompleteness* and what did Gödel's theorem say about it?
2. Describe Gödel's incompleteness theorem.
3. For a *consistent* system of logic, with arithmetic, what is a limit on what can be proven in the system? Give an example.
4. In your own words, what is asserted in the proof of Gödel's Incompleteness theorem?
5. Define *consistency* and *completeness*. What systems of logic have both?

5.3 Explain how recursion captures computability (priority)

1. Explain the relationship between primitive recursion and algorithmic computability.
2. Define the *zero* function and relate to primitive recursion.
3. Define the *successor* function and relate to primitive recursion.
4. Distinguish *primitive recursion* from *composition*.
5. Define the set of *projection* functions and relate to primitive recursion.
6. Define the *primitive-recursion* operation.
7. Describe how the logarithm function might be obtained from subtraction and division by primitive recursion.
8. What proof approach might show that a certain set of Java programs is equivalent to the μ-recursive functions? Describe.
9. How is the algorithmic notion of *repetition* implemented in recursive function theory? Give an example.
10. Explain the notion of *composition* in recursive function theory, and tell why we can say that the primitive-recursive functions are *closed under composition*.

Use operations on functions to show that the following are μ-recursive

11. multiplication
12. subtraction
13. division
14. exponentiation
15. logarithm
16. finding the smallest $x > 5$, such that x^3 is odd

5.4 Prove that a problem is undecidable

1. Compare and contrast the diagonal proofs by Cantor (cardinality of reals is greater than that of natural numbers) and Turing (some problems are uncomputable). What basic proof method do they share?
2. Briefly explain why no program or algorithmic machine can solve the halting problem.
3. A program is *correct* if it satisfies the program's specification. Is it possible to write a program that determines the correctness of another program? Explain.
4. Describe the program *S* that is used as a counter-example in the proof of undecidability of the halting problem.
5. What is wrong with the following? "It is easy to solve the Halting Problem. Just compile the code in question and see if it halts. If it does, output 'yes', otherwise 'no'."

5.5 Define a non-well-founded set coinductively

Using set notation, define of the set of infinite sequences of

1. bits
2. decimal digits
3. truth values
4. symbols chosen from the alphabet *A*
5. pairs of bits.
6. pairs of symbols from alphabet Σ.
7. input pairs of integers (x_1, x_2) and output values *y*.
8. symbols from alphabet Σ.

Consider the set of infinite sequences of inputs and outputs, when inputs are pairs of strings of symbols in the set DIGITS ('0' .. '9'), and outputs are strings of DIGITS.

9. Formally define the set of output strings.
10. How many different input pairs exist?
11. How many different output strings?
12. Formally define the set of *infinite sequences of* input/output pairs of natural numbers.
13. Formally define the set of *infinite sequences of* input pairs and output strings.

Multiple-choice questions on Topic 6 (Combinatorics)

1. Combinatorics and counting

1. A possibility tree diagrams (a) the likelihood of one outcome; (b) a series of events, each with *n* possible outcomes; (c) one event with *n* outcomes; (d) a linear series of events and outcomes; (e) none of these
2. By the Multiplication Rule, a series of *k* events, each with *n* possible outcomes, has ____ paths through its possibility tree (a) 1; (b) *k*; (c) *n*; (d) n^k; (e) k^k
3. A four-character PIN number, with 36 possibilities for each character, has ____ possible values (a) 4; (b) 36; (c) 4^{36}; (d) 36^4; (e) 36!
4. For finite disjoint sets *A* and *B*, $|A \cup B|$ = (a) $|A| + |B|$; (b) $\max\{|A|, |B|\}$; (c) $|A \cup B|$; (d) $|A|\,|B|$; (e) $|A| + |B| - |A \cap B|$
5. The Pigeonhole Principle states that if $|A| > |B|$ then (a) $f : A \rightarrow B$ is bijective; (b) $f : A \rightarrow B$ is surjective; (c) $f : A \rightarrow B$ is injective; (d) $f : A \rightarrow B$ is not injective; (e) $f : A \rightarrow B$ is not surjective
6. The assertion that, if $|A| > |B|$ then no injection from *A* to *B* exists, is called (a) inconsistency; (b) incompleteness; (c) uncountability; (d) undecidability; (e) the Pigeonhole Principle
7. The possible orderings of elements of a set are (a) truth values; (b) numbers; (c) sets; (d) combinations; (e) permutations
8. The possible unordered selections from a set are (a) truth values; (b) numbers; (c) sets; (d) combinations; (e) permutations
9. Permutations are ___ of a set (a) the elements; (b) the possible orderings of elements; (c) the sizes of subsets; (d) the subsets; (e) ways to describe elements
10. There are ____ permutations for *n* objects taken *k* at a time (a) *n*; (b) *n*!; (c) $(n - k)! / n!$; (d) $n! / (n - k)!$; (e) $n! / ((n - k)!\, k!)$
11. ___ are ordered (a) permutations; (b) combinations; (c) sets; (d) subsets; (e) none of these
12. Combinations are ___ of a set (a) the elements; (b) the possible orderings of elements; (c) the sizes of subsets; (d) the subsets; (e) ways to describe elements
13. Combinations are expressed as (a) $C(n, k)$; (b) n^k; (c) *n*!; (d) $n! / k!$; (e) k^k
14. There are ____ combinations for *n* objects taken *k* at a time (a) *n*; (b) *n*!; (c) $(n - k)! / n!$; (d) $n! / (n - k)!$; (e) $n! / ((n - k)!\, k!)$
15. ___ are unordered (a) permutations; (b) combinations; (c) sequences; (d) hierarchies; (e) none of these
16. $C(n, k)$ is also known as (a) permutations; (b) binomial coefficients; (c) Stirling numbers; (d) factorials; (e) a multiset
17. A multiset is a(n) (a) permutation; (b) ordered set; (c) *r*-combination with repetition allowed; (d) expression of probability; (e) exponential expression
18. An *r*-combination with repetition allowed is a (a) permutation; (b) ordered set; (c) multiset; (d) random variable; (e) expression of probability

2. Intractability

1. Intractable problems (a) are undecidable; (b) lack acceptable approximate versions; (c) take an unacceptably long time to solve; (d) lack solutions; (e) are easily solved
2. Exponential time is closely associated with (a) tractability; (b) combinatorial explosion; (c) constraint problems; (d) sorting problem; (e) interaction
3. AI problems tend to involve (a) computations with large numbers; (b) combinatorial explosion of running time; (c) easy choices once understood; (d) straightforward inference; (e) none of these
4. Deciding whether a formula in propositional logic is satisfiable is considered (a) intractable; (b) undecidable; (c) tractable; (d) decidable; (e) polymorphic
5. SAT is the problem of deciding whether a formula in propositional logic (a) holds; (b) has a set of variable assignments that make it true; (c) is not a contradiction; (d) is syntactically correct; (e) is probably true
6. The set of formulas in propositional logic that can evaluate to true values under some set of variable assignments is (a) SAT; (b) finite; (c) undecidable; (d) decidable in $O(n)$ time; (e) none of these
7. *P* is the set of (a) algorithms that execute in $O(n)$ time; (b) problems decidable in $O(n^k)$ time for some constant *k*; (c) problems *not* decidable in $O(n^k)$ time; (d) intractable problems; (e) exponential-time problems
8. Problems for which no polynomial-time solutions are known are called (a) undecidable; (b) intractable; (c) NP; (d) optimization; (e) none of these
9. NPC is the set of *all* (a) algorithms that execute in $O(2^n)$ time; (b) problems decidable in $O(n^k)$ time for some constant *k*; (c) problems for which possible solutions may be checked in $O(n^k)$ time; (d) intractable problems; (e) exponential-time problems
10. Problems to which *SAT* or similar problems are *reducible* are called (a) *P*; (b) *NP*; (c) *NP*-complete; (d) *NP*-hard; (e) undecidable
11. *NP*-complete problems are widely believed to have (a) polynomial-time solutions; (b) no polynomial-time solutions; (c) no exponential-time solutions; (d) no solutions checkable in polynomial time; (e) none of these
12. The set of intractable problems is associated with (a) *P*; (b) divide-and-conquer algorithms; (c) greedy algorithms; (d) *NP*; (e) *NPC* and *EXPTIME*

3. Discrete probability

1. A set of possible outcomes is a(n) (a) random variable; (b) probaiblity distribution; (c) compound event; (d) sample space; (e) permutation
2. An outcome that is from a set of uncertain possibilities characterizes a (a) random process; (b) sample space; (c) event; (d) sequence; (e) permutation
3. A set of possible outcomes is a(n) (a) random variable; (b) probability distribution; (c) compound event; (d) sample space; (e) permutation
4. An outcome that is from a set of uncertain possibilities characterizes a (a) random process; (b) sample space; (c) event; (d) sequence; (e) permutation

5. A uniform probability function P(x), for a probability space of size n, is (a) 0; (b) 1.0; (c) sqrt(n); (d) 1/n; (e) n
6. A probability space is (a) an event; (b) a random process; (c) a set of possible outcomes; (d) a random variable; (e) a set of probabilities
7. For sample space S, Kolmogorov's axiom asserts that P(S) = (a) 0; (b) 0.5; (c) 1; (d) 2; (e) indeterminate
8. For sample space S, Kolmogorov's axiom asserts that P($\varnothing$) = (a) 0; (b) 0.5; (c) 1; (d) 2; (e) indeterminate
9. Kolmogorov's axioms are considered useful for decision making because (a) they predict outcomes in many domains; (b) beliefs that violate the axioms result in poor bets; (c) they help the agent prove theorems; (d) they dictate inferences; (e) they reflect expertise
10. *Monotonicity* asserts that for probability spaces A and B, (a) $A \subseteq B \Rightarrow P(A) \leq P(B)$; (b) $A \subseteq B \Rightarrow P(A) = P(B)$; (c) $A = B \Rightarrow P(A) < P(B)$; (d) $A \subseteq B \Rightarrow P(A) \geq P(B)$; (e) none of these
11. P(~A) = (a) 0; (b) 1.0; (c) P(A); (d) 1 – P(A); (e) 1 / P(A)
12. For disjoint events A and B (a) $P(A \cap B) \geq 0$; (b) P(A) ≤ P(B); (c) $P(A \cup B) = P(A) - P(B)$; (d) P($A$) + P($B$); (e) $P(A \cup B) = 1.0$
13. For independent events A and B, $P(A \cap B)$ = (a) P(A) + P(B); (b) P(A) – P(B); (c) P(A) P(B); (d) P(A) / P(B); (e) 1.0
14. For independent events A and B, $P(A \cup B)$ = (a) P(A) + P(B) – P(~A) P(~B); (b) P(A) – P(B); (c) P(A) P(B); (d) P(A) / P(B); (e) 1.0
15. The average of values for equally likely outcomes is a(n) (a) probability; (b) random variable; (c) expected value; (d) combination; (e) permutation
16. Expected value of a die throw is (a) 0; (b) 1; (c) 3.5; (d) 4; (e) 6
17. Expected value of a coin toss is (a) 0; (b) 0.25; (c) 0.5; (d) 1; (e) 2
18. P(A | B) = (a) $P(A \cap B)$ / P(B); (b) $P(A \cup B)$; (c) P(A) P(B); (d) P(A) / P(B); (e) P(A) – P(B)
19. The average of values for equally likely outcomes is a(n) (a) probability; (b) random variable; (c) expected value; (d) combination; (e) permutation
20. Conditional probability is expressed by (a) P(A) + P(B) – P(~A) P(~B); (b) P(A) – P(B); (c) P(A) P(B); (d) P(A) / P(B); (e) P(A | B)
21. Conditional probability is (a) degree of belief in the absence of other information; (b) unconditional probability; (c) degree of belief given other information; (d) probability of a past event; (e) a random distribution
22. A degree of belief, in the absence of helpful information, is (a) prior probability; (b) conditional probability; (c) a random variable; (d) an axiom; (e) an event
23. A degree of belief given some helpful information is a(n) (a) prior probability; (b) conditional probability; (c) random variable; (d) axiom; (e) event
24. A random variable is a(n) (a) truth value; (b) set; (c) function; (d) relation; (e) number
25. A discrete random variable maps from (a) a sample space to [0..1]; (b) a sample space to a sample space; (c) a sample space to a number of outcomes; (d) outcomes to [0..1]; (e) outcomes to a sample space
26. A probability distribution maps from (a) a sample space to [0..1]; (b) a sample space to a sample space; (c) a sample space to a number of outcomes; (d) outcomes to [0..1]; (e) outcomes to a sample space
27. A random distribution takes values as follows (a) P(x) = k; (b) $P(x = k) = P(\{s \in S \mid \chi(s) = k\})$; (c) $f : S \rightarrow [0..1]$; (d) $P(A \cup B) = P(A) + P(B)$; (e) $P(x) \in [0..1]$
28. The normal curve depicts (a) the uniform distribution; (b) the Bayesian theorem; (c) the Gaussian distribution; (d) a random variable; (e) an outcome
29. A flat graph of a function depicts (a) the uniform distribution; (b) the Bayesian theorem; (c) the Gaussian distribution; (d) a random variable; (e) an outcome
30. Prior probability is (a) conditional probability; (b) unconditional probability; (c) degree of belief given other information; (d) probability of a past event; (e) a random distribution
31. Conditional probability may apply if events are (a) causal; (b) noncausal; (c) independent; (d) dependent; (e) identical
32. Conditional probability may apply if events are (a) causal; (b) noncausal; (c) independent; (d) dependent; (e) identical
33. Prior probability is (a) belief; (b) certainty; (c) conditional probability; (d) unconditional probability; (e) none of these
34. Probabilities of different event outcomes are a(n) (a) event; (b) probability distribution; (c) expected value; (d) sample space; (e) compound event
35. Any probability value is (a) 0 or 1; (b) in the range of 0 to 1; (c) some positive real number; (d) some positive or negative real number; (e) an integer
36. A sample space is (a) a random variable; (b) a sequence; (c) a number; (d) a set of all possible outcomes; (e) an event
37. Prior probability is (a) belief; (b) certainty; (c) conditional probability; (d) unconditional probability; (e) none of these

4. Bayes' theorem

1. Bayes' Theorem states that for hypotheses h and evidence E, (a) P(h_i) = P(E | h_i) P(h_i) / P(E); (b) P(h_i | E) = P(E | h_i) P(h_i) / P(E); (c) P(E) = P(E | h_i) P(h_i) / P(E); (d) P(h_i) = P(E | h_i) / P(E); (e) P(E) = P(E | h_i) / P(E);
2. Bayes' Theorem enables computation of probabilities of causes, given probabilities of (a) effects; (b) other causes; (c) prior world knowledge; (d) inference rules; (e) none of these
3. Evidence, in Bayes' Theorem, is (a) effects; (b) other causes; (c) prior world knowledge; (d) inference rules; (e) none of these
4. Bayes' Theorem is used in constructing (a) automata; (b) belief networks; (c) semantic networks; (d) knowledge bases; (e) none of these
5. ___ enables finding probabilities of causes, given effects (a) Minimax; (b) Bayes' Theorem; (c) Gödel's Theorem; (d) fuzzy logic; (e) Prolog

5. Other applications

1. Evolutionary computation uses the technique of maximizing (a) fitness; (b) reward; (c) performance; (d) quantity of output; (e) none of these
2. Evolutionary computation (a) is deterministic; (b) seeks optimal solutions; (c) was developed in the 19th century; (d) is probabilistic; (e) none of these
3. Evolutionary computation is modeled on (a) brute force; (b) divide and conquer; (c) greediness; (d) natural selection; (e) fractals
4. Function optimization searches for (a) a function; (b) parameter values; (c) a return value; (d) an algorithm; (e) a time analysis
5. Fitness measures are (a) parameters to functions; (b) functions to be optimized; (c) return values; (d) algorithms; (e) time functions
6. Evolutionary computation is (a) a brute-force method; (b) state-space search one state at a time; (c) path optimization; (d) population based; (e) DNA computing
7. Probabilities are employed in ____ methods (a) stochastic; (b) logical; (c) adversarial; (d) Java; (e) none of these
8. Modal logic has operators that reflect (a) certainty; (b) truth; (c) belief; (d) cost; (e) time
9. Degree of belief is expressed using (a) calculus; (b) logic; (c) probability theory; (d) temperature; (e) coin flipping
10. Stochastic methods are used in ___ reasoning (a) inferential; (b) diagnostic; (c) algorithmic; (d) paradoxical; (e) diagonal

Terminology for topic 6 (Combinatorics and probability)

atomic event
Bayes' Theorem
binomial coefficient
binomial distribution
binomial theorem
bounded rationality
combination
combinatorics
complementary event
complexity class
complexity of a problem
compound event
conditional probability
discrete probability
event
evolutionary computation
expected value
exponential time
independent events
intractable problem
Kolmogorov's axioms
Markov decision process
monotonicity
multiset
NP completeness
partial additivity
permutation
pigeonhole principle
possibility tree
prior probability
probability density function
probability distribution
probability of an event
random process
random variable
randomized algorithm
rational decisions
sample space
SAT
satisfiability
state space
stochastic methods
uniform distribution
uniform probability space

Questions to assess outcomes for topic 6

6.1 Solve a problem in permutations and combinations (priority)

1. What is the expected value of the roll of two dice, and why? Three dice? Four?
2. What is the expected number of heads in two coin tosses? Three? Four? Five?

Showing your work, give the (a) sample space and (b) probability that

3. Exactly one coin toss, of four, is a tail
4. At least two of four coin tosses are heads
5. The children in a four-child family are all girls (assume equal probability of boys and girls).
6. A seven-game world series will be swept in four games by one team or the other (assume evenly matched teams).

6.2 Describe the relationship between combinatorics and intractable problems

1. Define the two main complexity classes that distinguish tractable and intractable problems. Describe associated complexity classes.
2. What sorts of running times is *intractability* associated with, and why?
3. Describe the relationship among the following: combinatorial explosion; $O(n^k)$; $\Omega(2^n)$
4. Distinguish the problems of *validity* and *satisfiability* of propositional-logic formulas, referring to problem specification and complexity.
5. Concerning the following formula in propositional logic $(p \vee q \vee r) \wedge (p \vee \neg q \vee r) \wedge (\neg p \vee \neg q \vee \neg r)$:
   (a) State whether the formula is satisfiable, showing your work (you may write a truth table);
   (b) State and explain what is the time necessary to answer the question for arbitrary formulas with k variables.
6. What are the complexities of these problems w.r.t. formulas ϕ in propositional logic? Explain.
   a. ϕ is a tautology
   b. ϕ is satisfiable
   c. ϕ is a contradiction
   d. ϕ holds for a given set of variable assignments
7. What is $TIME(T(n))$?
8. What is the (very short) name of the set of problems that are decidable in time that is a polynomial function of the input size?

6.3a Describe a basic concept of probability theory (priority)

Define the following and give an example.

1. sample space
2. event
3. conditional probability
4. independent events
5. expected value
6. uniform distribution
7. random variable
8. atomic event

6.3b Prove a theorem in probability theory (priority)

Prove:

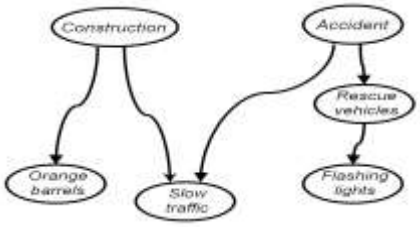
1. Monotonicity: $A \subseteq B \Rightarrow P(A) \leq P(B)$
2. $P(\neg A) = 1 - P(A)$
3. $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
4. $P(A \cup A^c) = 1$ (from Kolmogorov's axioms)
5. $P(A^c) = 1 - P(A)$ (from Kolmogorov's axioms)
6. (Uniform probability function $P : S \rightarrow$ **R)**
   $P(x) = (1/n)$ for any x in S
7. If A, B are disjoint events, then $P(A \cup B) = P(A) + P(B)$

6.4 Describe and apply Bayes' Theorem

1. Suppose $P(A \mid B) = 0.8$, $P(A) = 0.2$, and $P(B \mid A) = 0.3$. Give $P(B)$, using Bayes' Theorem[2], showing your work.
2. Describe how Bayes' Theorem is used to find quantitative predictions about cause-effect situations.
3. How does *conditional probability* enable diagnostic reasoning? Refer to Bayes' Theorem.
4. Express Bayes' Theorem in mathematical notation, based on the word description footnoted below.

[2] Bayes' Theorem states that the conditional probability of a hypothetical explanation for an observed event, given the event, is the product of unconditional probability of the hypothesis and the conditional probability of the event, given the hypothetical explanation, divided by the unconditional probability of the event.

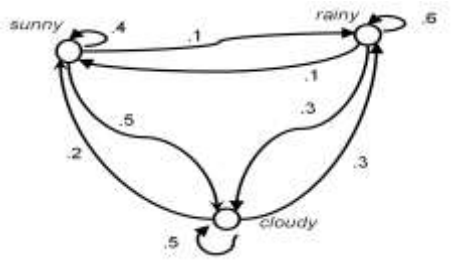
5. Name and describe the following, including the significance of possible labels on transitions.



6.5 Describe a computational application of probability theory

1. In your own words, relate *fitness* to *function optimization*.
2. What sorts of problems does evolutionary computation address and how?
3. Describe a population-based, randomized way to solve optimization problems by testing fitness.
4. Describe the evolutionary algorithm.

For #5-13, see the Markov model of weather below.



Based on that data, given that it is sunny today, showing your work, calculate the probability that:

5. The next three days will not all be sunny
6. The next three days will be cloudy
7. The next three days will be sun, clouds, rain
8. The next three days will be rainy
9. The next two days will be rain, then sun
10. Two of the next three days will be rainy
11. The next four days will all be sunny or cloudy
12. It will be cloudy two days from now
13. It will *not* be sunny two days from now

Multiple-choice questions on Topic 7 (Information theory)

1. Quantifying information

1. The mathematical framework of information theory is similar to (a) probability theory; (b) graph theory; (c) analysis of algorithms; (d) theory of computation; (e) entropy in statistical mechanics
2. The study of sources, channels, and outputs of communication systems is (a) information theory; (b) probability theory; (c) graph theory; (d) analysis of algorithms; (e) theory of computation
3. An application of information theory is (a) algorithm design; (b) algorithm analysis; (c) data compression; (d) models of computation; (e) function optimization
4. A concern of information theory is (a) excessive speed at source; (b) inefficiency at destination; (c) noise in channel; (d) broken connections; (e) algorithm design
5. Quantity of information in a message rises with (a) data speed; (b) accuracy of knowledge at source; (c) prior uncertainty about message; (d) prior certainty about message; (e) processor speed
6. Freedom of sender's choice of message determines (a) quantity of information content; (b) data speed; (c) accuracy of knowledge at source; (d) receiver's freedom; (e) processor speed
7. Entropy is (a) running time; (b) quantity of information; (c) certainty; (d) speed; (e) accuracy
8. Quantity of information is (a) entropy; (b) accuracy; (c) certainty; (d) speed; (e) order
9. A concern in data compression algorithm design is (a) increase of redundancy; (b) increase of information content; (c) decreased redundancy; (d) decreased information content; (e) line speed
10. Statistical properties of ergodic processes (a) vary throughout the processes; (b) are uniform throughout the processes; (c) are unrelated to entropy; (d) are indeterminate; (e) are simple
11. Entropy of a source depends on (a) speed of source; (b) probabilities of occurrence of symbols from the source; (c) probability of a given message; (d) probabilities of all messages; (e) none of these
12. The information in a random event E is ____ $P(E)$ (a) directly related to; (b) inversely related to; (c) directly related to the logarithm of; (d) inversely related to the logarithm of; (e) none of these
13. The information in a coin flip is ____ bits (a) 0; (b) 1; (c) 0.5; (d) 2; (e) 4
14. The information in two coin flips is ____ bits (a) 0; (b) 1; (c) 0.5; (d) 2; (e) 4

2. Algorithmic definitions of randomness

1. A string that is hard to compress is highly (a) simple; (b) random; (c) valued; (d) shrinkable; (e) improbable
2. The Kolmogorov complexity function quantifies (a) simplicity; (b) non-randomness; (c) compressibility; (d) algorithm running time; (e) incompressibility
3. If complexity $C(x) \leq |x| - c$, then x is (a) random; (b) c-compressible; (c) c-incompressible; (d) of high complexity; (e) of high running time
4. $K_M(x) = \min\{d \mid (\exists p)\ (|p| = d) \wedge \phi_M(p) = x\}$ is (a) a probability function; (b) a random variable; (c) a measure of complexity; (d) a measure of simplicity; (e) a measure of running time
5. The Invariance Theorem states that (a) complexity of strings is proportional to length; (b) complexity of strings is uniform; (c) complexity of strings is independent of language or model of computation; (d) all strings have the same amount of randomness; (e) none of these
6. Kolmogorov complexity if x (a) running time of a program x; (b) running time of a program that outputs string x; (c) length of the shortest program that outputs x; (d) the regularity of x; (e) the nonrandomness of x
7. The complexity of a string, $C(x)$, is (a) its length; (b) the number of different symbols in it; (c) its compressibility; (d) the length of the shortest algorithm that generates it; (e) the running time of the fastest-running algorithm that generates it
8. The length of the shortest program that outputs a string it the string's (a) compression ratio; (b) complexity; (c) algorithmic size; (d) time; (e) stature

3. Chaos and complex systems

1. The occurrence of seemingly random events in deterministic systems is (a) complexity; (b) probability; (c) combinatorics; (d) chaos; (e) fractal dimension
2. Chaos characterizes ____ systems (a) deterministic events in regular; (b) seemingly random events in deterministic; (c) random events in non-deterministic; (d) predictable events in simple; (e) predictable events in complicated
3. Aperiodic events characterize (a) clockwork; (b) randomness; (c) non-determinism; (d) chaos; (e) predictability
4. Chaos results from (a) internal processes; (b) linear feedback; (c) predictability; (d) nonlinear feedback; (e) algorithm execution
5. Nonlinear feedback results in ____ behavior (a) deterministic; (b) periodic; (c) predictable; (d) unpredictable; (e) time-consuming
6. Chaotic system behavior is heavily dependent on (a) processing speed; (b) human intervention; (c) initial conditions; (d) linear feedback; (e) none of these
7. Chaos is present when aperiodic behavior occurs in ____ systems (a) well-designed; (b) mathematically ornate; (c) mathematically simple; (d) very complicated; (e) none of these
8. Fractal geometry measures the ____ of objects (a) size; (b) regularity; (c) linearity; (d) irregularity; (e) form
9. Objects that are irregular all over and at the same degree at different scales are (a) linear; (b) simple; (c) fractals; (d) triangles; (e) circular
10. An *attractor* is a(n) (a) fractal; (b) complex system; (c) feedback generator; (d) initial state; (e) state to which a system settles
11. A state to which a chaotic system settles is called a (a) feedback loop; (b) complex result; (c) linear attractor; (d) strange attractor; (e) fractal

12. Self-organizing systems are (a) simple; (b) feedback-free; (c) open to their environments; (d) closed to their environments; (e) dependent on human control

13. Systems that are open, can maintain structure in non-equilibrium conditions, and complex in their feedback loops, are (a) strange; (b) self-organized; (c) fractals; (d) self-similar; (e) non-chaotic

Terminology for topic 7 (Information theory)

attractor
chaos
complex system
complexity
compressible
data compression
decentralization
descriptional complexity
emergent behavior
entropy
feedback
fractal geometry
incompressibility
information theory
invariance theorem
Kolmogorov complexity
linear feedback
nonlinear feedback
quantity of information
randomness
self-organization
strange attractor

Problems to assess outcomes for topic 7

7.1 Describe a way to quantify information (priority)

1. Describe a unit of measure used, in information theory, to quantify the information in a message.
2. What happens to the quantity of information in an image after JPEG compression of the image?
3. What is information theory?
4. *How much information* is contained in a string of a million bits, each of them 0?
5. What is entropy?
6. What principles guide data compression?
7. How is the amount of information in the outcome of an event related to the outcome's probability?
8. *How much information* is contained in a string of bits generated by a million coin flips?
9. What is Huffman code? How does the quantity of information in a string change after the string has been compressed using the Huffman scheme?

7.2 Define descriptional complexity and relate it to randomness

1. Relate *compressibility* to *randomness*.
2. Relate *complexity* to *randomness*.
3. What can be said about a string x, if complexity $K(x) \leq |x| - c$, for some constant c?
4. What is a c-incompressible string?
5. What is a way to quantify the *randomness* of an object?
6. What is *Kolmogorov complexity*?
7. Can the complexity of a string be defined only with respect to a particular machine or programming system, or may it be defined independently? Justify your answer.
8. If you flipped a coin 1000 times, how long do you think the shortest algorithm that generates the string would be?
9. If you used the Java random number generator to display a string x, a billion bits long, by simulating coin flips, what would you estimate the *complexity* of x to be?
10. Write a short algorithm to generate the string 0^{1000}. How much larger an algorithm would be required to generate the string $0^{1,000,000}$?

7.3 Relate chaos, complex systems, and self-organization

1. What is *chaos*?
2. Relate *aperiodic behavior* to *deterministic systems*.
3. What is *linear feedback* and how does it relate to chaos?
4. What is an *attractor*?
5. What is a *fractal*?
6. How do initial conditions influence the behavior of chaotic systems?
7. Give some features of self-organizing systems.

Projects

1. Find an online demo of chaotic or self-organizing systems.
2. Comment on M. Resnick's article about decentralized systems and decentralized thinking.
3. Download *Netlogo* and write up your experiences with it.

Study questions on multiple topics

1. Describe results obtained in this course by *induction* and results obtained by *contradiction*.
2. How did your ideas about the relationship between mathematics and computing change during this semester?
3. What recurrent threads appeared multiple times in this course?
4. What are *trees* and *recurrences* good for in computing applications?
5. What is a discrete structure? Give some examples.
6. Concerning discrete structures, how do you know what you know?
7. Give a practical application of a concept discussed in this course.
8. What is computing about?
9. Defend or refute: Discrete mathematical structures have a slight relationship with the problems encountered by IT professionals.
10. Referring to chapter __ of Epp,
    (a) What are the main concepts presented?
    (b) What are the main concepts presented in the topic(s) of this course for which the chapter was assigned?
    (c) Compare and contrast the textbook presentation with what was presented in the classroom and the slides.
11. Relate the seven topics discussed in this course. Focus on the way that some material occurred again and again in later topics. Contrast different approaches taken, problems addressed, and objects analyzed. Include discussion of ways to mathematically formalize some of the different notions addressed in the course. While covering all the main topics, you may emphasize ones that interested you the most.
12. How have your perspectives on the course material changed as a result of
    (a) group work,
    (b) comments on your work from other students,
    (c) comments from the instructor?
13. What ideas or passages in the textbook most engaged you or most changed the way you thought?
14. Describe some relationships among *logic*, *sets*, and *functions* as discussed in this course.
15. Discuss some knowledge that we have obtained in this course with mathematical certainty.
16. Explain what a *language* is and some mathematical notations for languages
17. Explain some operations on languages.
18. Describe some *finite* and *infinite* mathematical objects Distinguish among the main classes of infinite objects we discussed.
19. Discuss some mathematical structures that we have discussed that enable very fast solutions to certain problems
20. Discuss some tools that we have considered that enable us to see that certain problems are extremely time consuming to solve.
21. Describe at least three proof methods that we have used, together with examples of propositions proven by their use.
22. Discuss some mathematical ways that we have discussed of managing the uncertainty of the real world.
23. Discuss some proven limitations on our ability to solve certain problems at all.

Supplementary questions

T2: Sets

Fill blanks in proof: Epp, pp. 364-365, #2-5
Element argument: p. 365, #8, 10, 12-15, 17-20
Empty-set proof: p. 366, #25-35
Counter-example: p. 372, #1-4

Reflexive, symmetric, transitive relations: p. 458, #1-17
Equivalence relations: pp. 475-476, #1-19