CPSC 121 Some Sample Questions for the Final Exam
Tuesday, April 15, 2014, 8:30AM

Name: _____ Student ID: _____
Signature: _____ Section (circle one): George Steve
Your signature acknowledges your understanding of and agreement to the rules below.
(Note: our cover page will look different, but the rules will be the same.

– You have 150 minutes to write the 0 questions on this examination. A total of 0 marks are available.

– with any information you choose written or printed on them, as long as they are readable without magnification. **No other aids—including electronic devices—are allowed**; so, no cell phones and no calculators. (We will provide "Dave Tompkins's Awesome Handout", the predicate logic idioms sheet, and the "Proof Strategy Tips".)

– Keep your answers short. If you run out of space for a question, you have likely written too much.

– The number in square brackets to the left of the question number indicates the number of marks allocated for that question. Use these to help you plan your use of time on the exam.

– Clearly indicate your answer to each problem. If your answer is not in the provided blank, then indicate where the answer is, and at the answer's location indicate the question it addresses.

– **Good luck!**

Question	Marks
Total	

UNIVERSITY REGULATIONS:

– Each candidate should be prepared to produce, upon request, his/her UBC card.

– No candidate shall be permitted to enter the examination room after the expiration of one half hour, or to leave during the first half hour of the examination.

– CAUTION: candidates guilty of any of the following, or similar, dishonest practices shall be immediately dismissed from the examination and shall be liable to disciplinary action.

1. Having at the place of writing, or making use of, any books, papers or memoranda, electronic equipment, or other memory aid or communication devices, other than those authorised by the examiners.
2. Speaking or communicating with other candidates.
3. Purposely exposing written papers to the view of other candidates. The plea of accident or forgetfulness shall not be received.

– Candidates must not destroy or mutilate any examination material; must hand in all examination papers; and must not take any examination material from the examination room without permission of the invigilator.

[0] 1. **Tautologies and Contradictions:**

Determine whether the following statements are tautologies (definitely true), contradictions (definitely false), or contingencies (true or false depending on the values/definitions of the variables, domains, and predicates in the expressions). Indicate your answer by circling *one* of TAUTOLOGY, CONTRADICTION, and CONTINGENCY for each. (Use any technique you wish, truth tables, equivalences, proofs, etc.)

Assume all domains have at least one element (are non-empty).

[0] a. $(q \wedge r) \rightarrow F$

TAUTOLOGY CONTRADICTION CONTINGENCY

[0] b. $(a \vee b) \wedge ((a \wedge b) \rightarrow \sim c)$

TAUTOLOGY CONTRADICTION CONTINGENCY

[0] c. $\sim((q \wedge r) \rightarrow (\sim p \vee q))$

TAUTOLOGY CONTRADICTION CONTINGENCY

[0] d. $(p \vee q) \vee \sim(p \wedge q)$

TAUTOLOGY CONTRADICTION CONTINGENCY

[0] e. $\forall x \in \mathbf{S}, P(x) \rightarrow \exists y \in \mathbf{S}, P(y) \vee x \neq y$

TAUTOLOGY CONTRADICTION CONTINGENCY

[0] f. $\exists y \in \mathbf{S}, Q(y) \vee P(y)$

TAUTOLOGY CONTRADICTION CONTINGENCY

[0] g. $(\exists x \in \mathbf{S}, \exists y \in \mathbf{S}, x \neq y) \wedge (\exists y \in \mathbf{S}, \forall x \in \mathbf{S}, x = y)$

TAUTOLOGY CONTRADICTION CONTINGENCY

[0] h. $\forall x \in \mathbf{T}, \exists y \in \mathbf{S}, x = y$

TAUTOLOGY CONTRADICTION CONTINGENCY

[0] 2. **Quantifier Order and the Challenge Method:**

Consider the following theorem:

$$\exists x \in \mathbb{Z}, \forall y \in \mathbb{N}, ((\forall z \in \mathbb{Z}, P(x, y, z)) \wedge (\exists w \in \mathbb{N}, Q(x, y, w)))$$
$$\rightarrow \exists a \in \mathbf{A}, \forall b \in \mathbf{A}, R(x, y, a, b)$$

Imagine using a direct proof approach to prove this theorem (i.e., no contradiction, contrapositive, or other use of logical equivalences to alter the form of the theorem). For each quantifier, indicate whether you will **prove** the quantifier or **assume** it. If you are **proving** it, indicate whether you get to choose the variable's value and, if so, on the basis of which other variables. If you are **assuming** it, give an example of how you might make use of the assumption.

[0] 3. **Circuit Design:**

Design a circuit that, given inputs $I = i_0 i_1 i_2$ and p, determines the remainder when the signed binary number I is divided by the P^{th} prime number according to the table:

P	prime
0	2
1	3

The remainder should be an unsigned binary number of the minimum number of bits necessary.

For example, if $i_0 i_1 i_2 = 100$ and $p = 1$, then I is -4 and the prime is 3. The closest we can get to -4 divided by 3 while keeping the result no larger than -4 (so we avoid a negative remainder) is $3 * -2 = -6$. The remainder is 2. So, the output would be 10.

Provide propositional logic expressions for the circuit's output(s). **Do NOT draw the circuit.** Show your work for partial credit.

[0] 4. **Number Representation:**

The Java language specification has the following to say about its numeric and boolean types:

> The *integral types* are `byte`, `short`, `int`, and `long`, whose values are 8-bit, 16-bit, 32-bit and 64-bit signed two's-complement integers, respectively, and `char`, whose values are 16-bit unsigned integers representing UTF-16 code units (3.1).
>
> The *floating-point types* are `float`, whose values include the 32-bit IEEE 754 floating-point numbers, and `double`, whose values include the 64-bit IEEE 754 floating-point numbers.
>
> The `boolean` type has exactly two values: `true` and `false`.

The C++ language specification has the following to say about its integral types:

> There are five standard signed integer types : "signed char", "short int", "int", "long int", and "long long int". In this list, each type provides at least as much storage as those preceding it in the list. There may also be implementation-defined extended signed integer types. The standard and extended signed integer types are collectively called signed integer types. Plain ints have the natural size suggested by the architecture of the execution environment; the other signed integer types are provided to meet special needs.
>
> Unsigned integers, declared unsigned, shall obey the laws of arithmetic modulo 2^n where n is the number of bits in the value representation of that particular size of integer.

Now answer the following questions:

[0] a. The Java programming language was designed so that programs written in Java could be compiled on any type of computer and later run on any *other* type of computer that implements the "Java virtual machine" that runs Java programs. C++ was not. Discuss how these design targets affected the language specifications.

[0] b. Some Java and C++ programs test to see if `int` values in the program are negative by checking whether the 32nd bit from the right is 1. Based on the specifications (for each language), is this guaranteed to be correct? Why or why not?

[0] c. If a `long long int` on a particular computer takes 32 hexadecimal digits to represent, how many bits long is it?

[0] d. Roughly speaking, a "floating-point number" is like the "fixed-point" representation we learned plus scientific notation (i.e., multiplied by 2^k, where k is specified as part of the value). Do you think Java's `float` and `double` types can precisely represent $\frac{1}{11}$? Why or why not?

[0] e. If a Java `boolean` were actually represented in the computer using the same amount of space as a `byte`, how many "wasted bit patterns" would the representation have? (A "wasted bit pattern" is a pattern that means nothing or that has the same meaning as some other pattern already counted as not "wasted".)

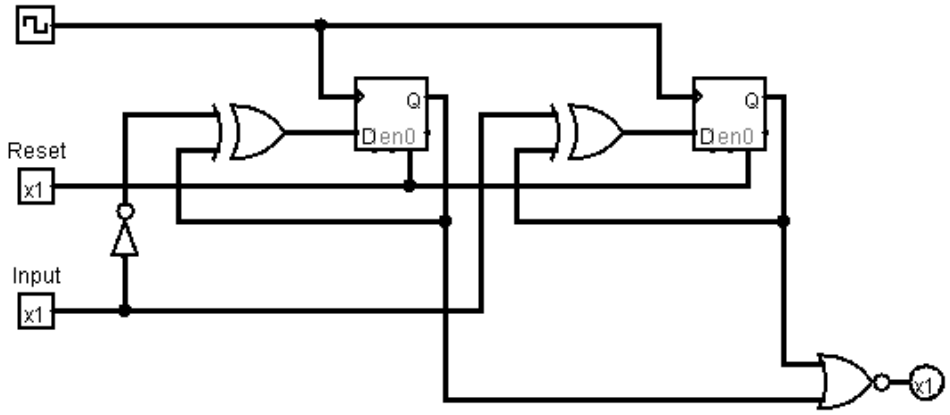
[0] 5. **Direct Proof Approaches:**

For the following theorem, give a "direct" proof approach—no proof by contradiction or contrapositive, no use of logical equivalences—that is as complete as possible. (If there are choices that the person using your approach will need to make, be sure to give as much detail as possible about how they can make those choices.)

$$\forall x_1 \in \mathbf{A}, \exists y_1 \in \mathbf{B}, P(x_1, y_1, 0) \wedge$$
$$\forall x_2 \in \mathbf{A}, \exists y_2 \in \mathbf{B}, \forall z_1 \in \mathbf{A}, P(x_2, y_2, z_1) \rightarrow \exists z_2 \in \mathbf{A}, Q(z_2, z_1) \wedge P(x_2, y_2, z_2)$$

[0] 6. **Lab Circuits:**

Consider the following circuit:



Assume that all of the flip-flops start with the value 0. Assume that "Input" is a sequence of bits provided by the user and that the clock ticks each time a new bit is available.

[0] a. Give three different input sequences of different lengths that all lead to outputs of 1.

[0] b. Give an input and a state for the circuit (i.e., a bit stored by the left flip-flop, a bit stored by the right flip-flop) such that when the clock ticks **both** flip-flops change their stored values **or** explain why no such input and state exists.

[0] c. **Briefly** explain in English the meaning of the left flip-flop's value.

[0] d. **Briefly** explain in English the meaning of the right flip-flop's value.

[0] e. **Briefly** explain in English what the circuit does.

[0] f. Modify the circuit so that it outputs 1 exactly when the number of 1s input to the circuit is divisible by 4.

[0] 7. **Predicate Logic:**

Much of Facebook's data is described as a "graph" (not the kind with x- and y-axes), which is made up of nodes and edges that connect nodes together. The edges express relationships between the nodes.

We'll reason about a similar graph using the following definitions:

- N is the set of all nodes
- $T = \{$"friend", "located in", "likes", "owns"$\}$ is the set of relationship types
- $\mathrm{Edge}(a, b, t)$ means that node a has the relationship t with node b. (Note that a may have the "likes" relationship with b while b does not have that relationship with a; so, order matters.)
- $\mathrm{Person}(a)$ means node a represents a person
- $\mathrm{Business}(a)$ means node a represents a business
- $\mathrm{Location}(a)$ means node a represents a location

For convenience, you may use E for Edge, P for Person, B for $\mathrm{Business}$, and L for $\mathrm{Location}$.

Defining Predicates:

Define the following predicates in predicate logic in terms of the predicates and sets given above.

[0] a. $\mathrm{Locale}(l)$ means l is a location and is not itself located in any other location.
$\mathrm{Locale}(l) \equiv$

[0] b. $\mathrm{Neighbors}(a, b)$ means nodes a and b are at the same locale. (You may assume every node is in at most one locale, but you should **not** assume that every node is in *exactly one* locale.)
$\mathrm{Neighbors}(a, b) \equiv$

Making Statements:

State the following facts in predicate logic. **YOU MAY** (but do not have to) **USE THE EXTRA PREDICATES YOU WERE ASKED TO DEFINE ABOVE.** (Assume they are correct.)

[0] a. Friendship is "symmetric"; that is, two people either have the friend relationship with each other or neither has the friend relationship with the other.

[0] b. Every node is a person, business, or location, but not more than one of these.

[0] c. A business owner's friends all like the business.

[0] 8. **Indirect Proof:**

Consider the theorem: Every DFA either accepts an infinite number of different inputs or rejects an infinite number of different inputs (or both).

Prove the theorem using proof by *contradiction* (and no other steps preceding the contradiction step).

[0] 9. **Sets and Functions:**

Consider the set $A = \{2, 3, 4, \ldots, 99\}$ and $B = \{true, false, unknown\}$.

[0] a. Is the empty set a member of A?

[0] b. Is the empty set a subset of B?

[0] c. Give a one-to-one (injective) function $f : A \to B$ or explain why no such function exists.

[0] d. Give an onto (surjective) function $f : A \to B$ or explain why no such function exists.

[0] e. Give a one-to-one (injective) function $f : B \to A$ or explain why no such function exists.

[0] f. Give an onto (surjective) function $f : B \to A$ or explain why no such function exists.

[0] g. What is $|P(A)|$? You can leave your answer as a mathematical expression of only numbers and arithmetic operators (including exponentiation if you need it).

[0] h. What is $P(B)$?

[0] i. In terms of A and B, what is the domain and codomain of a function f that takes a natural number greater than 1 and less than 100 and produces either (a) the Boolean value $true$ if the number is composite and the number's largest factor or (b) the Boolean value $false$ if the number is prime and the number itself. (For example, $f(2)$ produces $false$ and 2 while $f(28)$ produces $true$ and 14.)

[0] 10. **Designing DFAs:**

[0] a. Design a Deterministic Finite-State Automaton (DFA) that takes as input a sequence of bits—0 or 1—and accepts exactly those sequences that include at least three bits of which at least two are 0s. (So, for example, `000`, `001`, `010`, `100`, `11010`, and `1111011001011` should all be accepted. `00`, `110`, `1110111`, `1`, `0`, and the empty string should all be rejected.)

[0] b. Now, write a regular expression that matches the same set of strings.

[0] 11. **Direct Proof:**

Prove

$\forall a \in \mathbb{Z}^+, \forall b \in \mathbb{Z}^+, \exists n \in \mathbb{N}, n > b \wedge n^2 > an + 1.$

[0] 12. **Working Computer:**

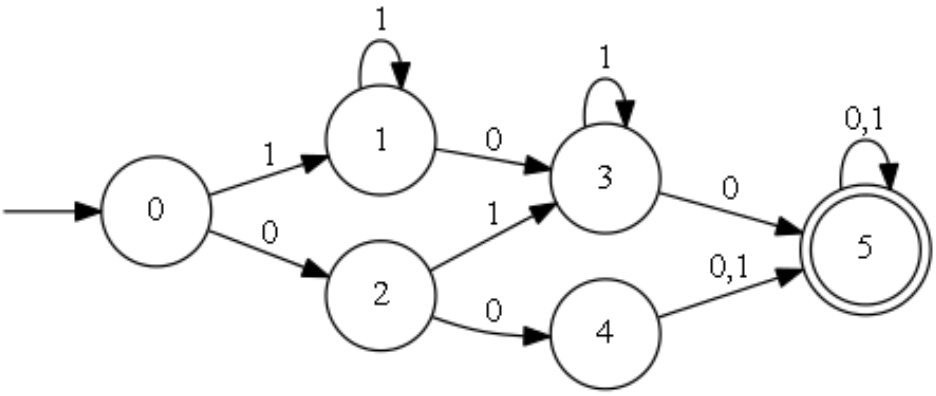
Our working computer's instructions can come in various lengths.

[0] a. Why are some instructions longer than others? Briefly explain, ideally with an example of what might cause an instruction to require more space.

[0] b. What part of an instruction indicates how much memory the instruction takes up?

[0] 13. **DFA Implementation:**

Consider the following DFA:



[0] a. What state will this be in after processing the string 11011?

[0] b. Would this DFA still be legal if we changed state 0 to be an accepting state?

[0] c. Would this DFA still be legal if we eliminated the arc from state 5 to itself?

[0] d. Implement this DFA as a circuit.

[0] 14. **Induction:**

For any positive integer n, the n^{th} Fibonacci number F_n is defined to be:

$$F_n = \begin{cases} 1 & \text{if } n = 1 \\ 1 & \text{if } n = 2 \\ F_{n-1} + F_{n-2} & \text{if } n > 2 \end{cases}$$

[0] a. Prove that the sum of two odd numbers is an even number. (For this and the next two problems: probably not by induction!)

[0] b. Prove that the sum of two even numbers is an even number.

[0] c. Prove that the sum of an odd number and an even number is an odd number.

[0] d. Prove **by induction** that every third Fibonacci number (and no others) is even. That is: F_n is even when n is divisible by 3 and odd otherwise. (*Hint*: draw your recursive structure from the definition of the Fibonacci numbers, divide into three cases, and use the facts you just proved above.)

[0] e. Prove **by induction** that twice the sum of the first k even Fibonacci numbers is equal to the sum of the first $3k$ Fibonacci numbers. (*Hint*: draw your recursive structure from the summation, **not** the definition of the Fibonacci numbers, and use the fact you just proved.)

[0] 15. **More Induction:**

Prove **by induction** that adding 1 to a binary number with b bits (using fixed bit-width, i.e., the result has b bits no matter what, so $111\ldots111+1=000\ldots000$) changes no more than one 0 bit to a 1 bit.

Hint: write out the process of adding 1 to a binary number. Two cases do not require adding 1 to another binary number; one does. Is this a recursive structure? Which cases are base cases? Which cases are recursive cases?

[0] 16. **Even More Induction:**

We still don't know what binomial queues are, but we can learn something about them!

A binomial queue is made up of many trees. The shape of the tree in the 0^{th} slot is a single node. The shape of the tree in the k^{th} slot is a "root" node with k subtrees, which have the same shapes as the trees in the 0^{th}, 1^{st}, 2^{nd}, $\ldots$, and $(k-1)^{\text{th}}$ slots.

[0] a. Prove **by induction** that $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$ (for all natural numbers n).

[0] b. Using the result from the previous part, prove **by induction** that there are 2^n nodes in the tree in the n^{th} slot in a binomial queue (for all natural numbers n).

[0] c. Based on the previous parts, prove (however you would like) that the sum of the number of nodes in the trees in the first k slots (slot $0, 1, 2, \ldots, k-1$) in a binomial queue is equal to $2^k - 1$ (for all natural numbers k).

[0] d. Prove (however you would like) that for all natural numbers k, the tree in the $(k+1)^{\text{th}}$ slot of a binomial queue has the same shape as the tree in the k^{th} slot except the root node has one additional subtree, which itself has the same shape as the tree in the k^{th} slot.

This page left intentionally (almost) blank.

If you write answers here (or anywhere other than their intended location), mark them clearly, indicate which question they respond to, and indicate at the provided solution blank for that question where you wrote your solution.