Final Exam

Read this page and fill in your name, pledge, and email ID now.

Name: _____

Email ID: _____

For this exam, you must **work alone**. You are not permitted to obtain help from people other than asking clarifying questions of the course staff. You are not permitted to provide help to others taking the exam. **You may not use any resources other than the one page of notes you prepared.**

Sign below to indicate that you understand these expectations and can be trusted to behave honorably:

Signed: _____

The exam has 12 questions. For each question, there is ample space provided to hold an excellent answer. If you need more space, you can use the backs of pages, but include clear markings and arrows to indicate the answer that should be graded. We will assume anything not inside the answer space or clearly marked from it, is your scratch work that should not be considered in scoring your answers.

Do not open past this page until instructed to do so.

Logical Formulas and Inference Rules

1. For each candidate inference rule below, indicate if it is *sound* or *unsound* (circle the correct answer). For the rules that are unsound, provide a counter-example to show it is unsound. You do not need to provide any justification for the rules that are sound.

   a. $\dfrac{P \wedge Q}{P}$

   Circle one: *Sound* *Unsound*
   Counter-example (if unsound):

   b. $\dfrac{P \wedge (Q \rightarrow P)}{Q}$

   Circle one: *Sound* *Unsound*
   Counter-example (if unsound):

   c. $\dfrac{(\overline{P} \vee Q) \wedge (\overline{Q} \vee R)}{\overline{P} \vee R}$

   Circle one: *Sound* *Unsound*
   Counter-example (if unsound):

   d. $\dfrac{(P \wedge Q) \wedge (P \vee Q)}{P \text{ XOR } Q}$

   Circle one: *Sound* *Unsound*
   Counter-example (if unsound):

Well Ordering

2. For each set and operator below, answer if the set is *well-ordered* or not. Support your answer with a brief, but clear and convincing, argument.

a. The set of integers between -100 and 100; $<$.

Circle one: *Well-Ordered* *Not Well-Ordered*
Justification:

b. The non-negative rational numbers; $<$.

Circle one: *Well-Ordered* *Not Well-Ordered*
Justification:

c. The set $P_S = \text{pow}(S)$ (i.e., the set of all subsets of S) where $S = \{1, 2, 3\}$; under the comparator: $\subseteq$.

Circle one: *Well-Ordered* *Not Well-Ordered*
Justification:

Revisiting Revisiting Exam 1

3. Recall this question that appeared on Exam 1 and Exam 2: *Explain why the set of real numbers x where $-1 \leq x \leq 1$ is not well ordered by "$<$".* For each of the candidate answers below, indicate if the answer is *Good*, *Fixable*, or *Hopeless*. If the answer is *Fixable*, explain concisely how to fix it. If the answer is *Hopeless*, explain why the approach used in the answer cannot reasonably lead to a good answer.

a. We prove using the well-ordering principle.
Define the set of counterexamples,

$$C = \{z \mid \text{the set of real numbers } z \leq 1 \text{ is not well-ordered by } <\}.$$

If C is non-empty, there exists a minimum $m \in C$.
By the definition of C, the set of real numbers $m \leq 1$ is not well-ordered by $<$. But, m is the minimum element of that set. So, we have a contradition. Thus, by the well-ordering principle the proposition must be true.

Circle one: *Good* or *Fixable* or *Hopeless*

Explanation (nothing required if *Good*; if *Fixable*, explain how to fix; if *Hopeless* explain why):

b. We prove by induction on the size of the set, S.

Base case: $|S| = 1$. The set has one element, so that element must be its minimum.
Induction case: by the induction hypothesis, we assume that all sets R where $|R| = |S| - 1$ have a minimum element, m_r, and show that S has a minimum element. So, there is some new element, x, such that $S = R \cup \{x\}$. The minimum element of S is either x, if $x < m_r$, or m_r otherwise. Thus, S has a minimum.

Circle one: *Good* or *Fixable* or *Hopeless*

Explanation: (nothing required if *Good*; if *Fixable*, explain how to fix; if *Hopeless* explain why)

c. Assume by contradiction that, R_1, the set of real numbers between -1 and 1 is well ordered. By the definition of a well-ordered set, all non-empty subsets of R_1 have a minimum element. Consider $S = R_1 - \{-1\}$. We know $S \subset R_1$ since every element of S is an element of R_1. We show that R_1 is not well-ordered by contradiction. By the assumption that R_1 is well ordered, since S is a non-empty subset of R_1 it must have a minimum, m. Define $m^* = (-1 + m)/2$. Then $m^* < m$ since $m > -1$ and the average of -1 and m must be greater than -1. But $m^* \in S$ since it is a real number between -1 and 1. This contradicts the assumption that m is the minimum of S. This contradicts the assumption that R_1 is well-ordered, since we have shown a nonempty subset of R_1 that has no minimum.

Circle one: *Good* or *Fixable* or *Hopeless*

Explanation (nothing required if *Good*; if *Fixable*, explain how to fix; if *Hopeless* explain why):

Sets and Relations

4. Indicate for each statement if it is valid (always true) or invalid. For invalid statements, provide a counter-example supporting your answer.

a. For any finite sets A and B, it holds that $|A| - |B| \leq |A \cap B|$.

Circle one: *Valid* *Invalid*
Counter-example (if invalid):

b. For any sets A, B, and C, if there is an injective ([≤ 1 arrow in]) function ([≤ 1 arrow out]) f from A to B and an injective function g from B to C, then there exists an injective function from A to C.

Circle one: *Valid* *Invalid*
Counter-example (if invalid):

c. If $A \in \text{pow}(B)$ and $B \in \text{pow}(C)$, then $A \in \text{pow}(C)$.

Circle one: *Valid* *Invalid*
Counter-example (if invalid):

Induction

5. Suppose A and B are finite sets. Prove by induction that $|A \times B| = |A| \cdot |B|$ where $A \times B$ is the cartesian product of A and B. (Hint: you can apply the induction over the size of A. A good answer must clearly define the induction predicate.)

Cardinality

6. For each set defined below, answer if the set is *"finite"*, *"countably infinite"*, or *"uncountable"* and support your answer with a convincing and concise proof. (Recall that $\mathbb{N}$ is the set of natural numbers, $\mathbb{R}$ is the set of real numbers.)

a. pow(pow(pow(S))) where S is the set of all students in cs2102 Fall 2017.

Circle one: *Finite* *Countably Infinite* *Uncountable*

Proof:

b. $\{r \mid r = y/2^x, x \in \mathbb{N}, y \in \mathbb{N}\}$

Circle one: *Finite* *Countably Infinite* *Uncountable*

Proof:

c. $\{(a,b) \mid a \in \mathbb{N}, b \in \text{pow}(\mathbb{N})\}$

Circle one: *Finite* *Countably Infinite* *Uncountable*

Proof:

d. $\{(a,b,c) \mid a \in \mathbb{N}, b \in \mathbb{N}, c \in \mathbb{N}\}$

Circle one: *Finite* *Countably Infinite* *Uncountable*

Proof:

Recursive Data Types

Consider the recursive data types, *XBF* (short for Xor-Based Formula) defined by:

– **Base:** *true* is an *XBF*.
– **Base:** *false* is an *XBF*.
– **Constructor:** for all *XBF* objects f_1, f_2, *xor*(f_1, f_2) is a *XBF*.

The Value of an *XBF*, f, should be the Boolean value of it when it is evaluated as a logical formula based on XOR gates. To distinguish the syntactic symbols used in defining *XBF* s from the logical Booleans, we use **True** and **False** to represent the Boolean true and false values, and $\oplus$ to represent the Boolean operation xor.

So, for example,

$$\text{Value}(\textit{xor}(\textit{xor}(\textit{true}, \textit{false}), \textit{true})) = \textbf{False}.$$

7. Provide a precise, sensible, and complete definition of Value for all *XBF* objects.

8. [Excluded from exam]

Number Theory

9. Indicate for each statement if it is True or False. Either way, provide a short reason supporting your answer.

a. For all positive natural numbers a, b, c, it holds that $\gcd(a, \gcd(b, c)) = \gcd(\gcd(a, b), c)$.

Circle one: *True* *False*
Short justification:

b. $\mathbb{N}_5$ is a field if we use $(+ \bmod 5)$ for addition and $(\times \bmod 5)$ for the multiplication operations.

Circle one: *True* *False*
Short justification:

c. There exist (distinct) primes $p < q < r$ such that $pr = q^2$.

Circle one: *True* *False*
Short justification:

Program Correctness

We call a list $p_0, \ldots, p_n$ a non-decreasing list, if $p_0 \leq p_1 \leq \ldots p_n$. Consider the Python program below, that returns `True` if and only if the elements in the input list are non-decreasing. We assume that `p` is a non-empty list of natural numbers.

```
def non_dec(p):
   i = 0
   q = p[0]
   g = True
   while i < (len(p) - 1):
      i = i + 1
      if p[i] < q:
         g = False
      q = p[i]
   return g
```

10. Complete the definition of the state machine, $M_g = (S, G, q_0)$, below that models `non_dec`.

$$
\begin{aligned}
S &= \{(i, q, g) \mid i \in \mathbb{N}, q \in \mathbb{N}, g \in \{\textbf{True}, \textbf{False}\}\} \\
G &= \{(i, q, g) \rightarrow (i', q', g') \mid \\
&\qquad i, i', q, q' \in \mathbb{N}, g, g' \in \{\textbf{True}, \textbf{False}\} \\
&\qquad \wedge\ i < \texttt{len(p) - 1} \\
&\qquad \wedge\ i' = \underline{\qquad\qquad\qquad\qquad} \\
&\qquad \wedge\ \underline{\qquad\qquad\qquad\qquad} = p[i'] \\
&\qquad \wedge\ g' = \begin{cases} \underline{\quad} & \text{if } p[i'] < q \\ \underline{\quad} & \text{otherwise} \end{cases} \\
&\qquad \} \\
q_0 &= \underline{\qquad\qquad\qquad\qquad}
\end{aligned}
$$

(This is repeated from the previous page, so you have it handy here.)

$$
\begin{aligned}
S &= \{(i,q,g) \mid i \in \mathbb{N}, q \in \mathbb{N}, g \in \{\textbf{True}, \textbf{False}\}\} \\
G &= \{(i,q,g) \to (i',q',g') \mid \quad i,i',q,q' \in \mathbb{N}, g,g' \in \{\textbf{True}, \textbf{False}\} \\
&\qquad \wedge\, i < \texttt{len(p) - 1} \\
&\qquad \wedge\, i' = \underline{\qquad\qquad\qquad\qquad} \quad \wedge \quad \underline{\qquad\qquad\qquad\qquad} = p[i'] \\
&\qquad \wedge\, g' = \begin{cases} \underline{\quad} \text{ if } p[i'] < q \\ \underline{\quad} \text{ otherwise} \end{cases} \\
&\qquad \} \\
q_0 &= \underline{\qquad\qquad\qquad\qquad}
\end{aligned}
$$

11. Prove that for any input that is a finite non-empty list of natural numbers, the state machine M_g always terminates, and the final state is a state where the value of g is **True** if and only if the input list is non-decreasing. Note that you need to do the following: (a) Prove that the program ends. (b) Formally define a property P and show that P for states of the machine above and prove that P is a preserved invariant. (c) Show that P holds for the initial state. (d) Show that if P holds for a final state, then that state will have the correct answer.

(This page is intentionally left almost blank. You can use it for work, or to continue your answer to #11.)

Optional Guidance

This question is optional, and it will not count against you if you decline to answer it.

What grade do you believe you deserve in cs2102? _____

Explain why:

Anything else you want us to know?

Score: _____

