Practice Second Midterm Exam V

This exam is closed-book and closed-computer. You may have a double-sided, 8.5” × 11” sheet of notes with you when you take this exam. You may not have any other notes with you during the exam. You may not use any electronic devices (laptops, cell phones, etc.) during the course of this exam. Please write all of your solutions on this physical copy of the exam.

You are welcome to cite results from the problem sets or lectures on this exam. Just tell us what you're citing and where you're citing it from. However, please do not cite results that are beyond the scope of what we've covered in CS103.

On the actual exam, there'd be space here for you to write your name and sign a statement saying you abide by the Honor Code. We're not collecting or grading this exam (though you're welcome to step outside and chat with us about it when you're done!) and this exam doesn't provide any extra credit, so we've opted to skip that boilerplate.

You have three hours to complete this practice midterm. There are 36 total points. This practice midterm is purely optional and will not directly impact your grade in CS103, but we hope that you find it to be a useful way to prepare for the exam. You may find it useful to read through all the questions to get a sense of what this practice midterm contains before you begin.

You have three hours to complete this exam. There are 48 total points.

Question	Points	Graders
(1) Equivalence Relations	/ 12	
(2) Strict Orders	/ 12	
(3) Graph Theory	/ 12	
(4) Induction and Cardinality	/ 12	
	/ 48	

Problem One: Equivalence Relations (12 Points)

(Midterm Exam, Fall 2017)

On Problem Set Three, you explored equivalence relations and equivalence classes. On Problem Set Five, you saw how to apply the pigeonhole principle to binary relations. This problem is designed to let you show what you've learned in the course of doing so.

As a refresher, if R is an equivalence relation over a set A and $x \in A$, then the ***equivalence class of x in R***, denoted $\mathbf{[x]_R}$, is the set

$$[x]_R = \{ \ y \in A \ \mid \ xRy \ \}.$$

A foundational result about equivalence relations is the ***fundamental theorem of equivalence relations***, which says that if R is an equivalence relation over a set A and $x \in A$, then x belongs to exactly one equivalence class in R. Although we stated this result in class, we never actually proved it. As a first step in this problem, we'd like you to prove a part of this result.

i. **(4 Points)** Prove that if R is an equivalence relation over a set A and $x \in A$, then x belongs to ***at least*** one equivalence class in R. To clarify, you do ***not*** need to prove that x belongs to *exactly* one equivalence class in R, just that it belongs to *at least* one equivalence class in R.

   Feel free to use this space for scratch work. There's room to write your answer on the next page of this exam.

(Extra space for your answer to Problem One, Part (i), if you need it.)

Let's now introduce some new notation. If R is an equivalence relation over a set A, the ***index*** of R, denoted ***I*(*R*)**, is the number of equivalence classes in R. Additionally, the ***width*** of R, denoted **W(*R*)**, is the number of elements in the largest equivalence class in R.

ii. **(8 Points)** Using your result from part (i) and the pigeonhole principle, prove that if R is an equivalence relation over a set A and $|A| = n^2 + 1$ for some positive natural number n, then $I(R) \geq n + 1$ or $W(R) \geq n + 1$ (or both).

Feel free to use this space for scratch work. There's room to write your answer on the next page of this exam.

(Extra space for your answer to Problem One, Part (ii), if you need it.)

Problem Two: Binary Relations (12 Points)

(Midterm Exam, Fall 2017)

In C, C++, Java, and many other languages, the `double` type represents a real number, with a few key caveats. One particular example of this is the special value **`NaN`**, short for "not a number." In C++, if you try to take the square root of a negative number, the result is `NaN`. (You can also get `NaN` by dividing 0.0 by 0.0, and in a few other ways). For example:

```
double result = sqrt(-137.0);
cout << result << endl;        // Prints NaN
```

Notice that the value `NaN` is actually stored in the variable `result`. `NaN` is a perfectly legal value for a `double`, just as 137 and -2.718 are.

Things get a bit weird when looking at comparisons between `double`s when one or both `double`s is `NaN`. Specifically, *any comparison* involving `NaN` will evaluate to false. For example, in C++, all of the following comparisons evaluate to false:

`NaN < 137`   `137 < NaN`   `NaN < NaN`

Modeling things mathematically, we can imagine that the `double` type represents either a real number or the special value `NaN`. In other words, a `double` represents a value from $\mathbb{R} \cup \{\texttt{NaN}\}$. The less-than operator over `double`s in C++ then behaves like the following relation S over the set $\mathbb{R} \cup \{\texttt{NaN}\}$:

$$xSy \qquad \text{if} \qquad x \neq \texttt{NaN} \;\land\; y \neq \texttt{NaN} \;\land\; x < y$$

(In the above, the notation $x < y$ means that x is less than y in the conventional sense of one real number being less than another. Also, remember that the "if" in the above statement is interpreted as "is defined as" and is not an implication.)

i. **(5 Points)** Prove that S is a strict order over the set $\mathbb{R} \cup \{\texttt{NaN}\}$. You can assume that the $<$ relation over $\mathbb{R}$ is a strict order.

   Feel free to use this space for scratch work. There's room to write your answer on the next page of this exam.

(Extra space for your answer to Problem Two, Part (i), if you need it.)

As a refresher from the previous page, the relation S is defined over the set $\mathbb{R} \cup \{\texttt{NaN}\}$ as follows:

$$xSy \qquad \text{if} \qquad x \neq \texttt{NaN} \;\land\; y \neq \texttt{NaN} \;\land\; x < y$$

Again, the notation $x < y$ denotes the less-than relation over real numbers, which you can assume is a strict order and behaves the way you've seen it work in the past.

On Problem Set Three, you explored how, for a strict order R over a set A, it's possible to define the ***incomparability relation*** of R, denoted $\boldsymbol{\sim_R}$. As a refresher, the incomparability relation $\sim_R$ is a binary relation over the set A defined as follows:

$$x \sim_R y \qquad \text{if} \qquad x \not R y \land y \not R x$$

Notice that there are slashes through those R's, and remember that the "if" here means "is defined as."

ii. **(3 Points)** Give a definition for the $\sim_S$ relation over $\mathbb{R} \cup \{\texttt{NaN}\}$ by filling in the following blank. Please provide the simplest answer that you can. No justification is necessary. First-order logic notation is perfectly fine here.

$x \sim_S y$ if ____

Below is some scratch space you can use in the course of working through this problem.

As you saw on Problem Set Three, a binary relation R over a set A is called a ***strict weak order*** if R is a strict order and the relation $\sim_R$ is transitive.

iii. **(4 Points)** Prove or disprove: the S relation over $\mathbb{R} \cup \{\texttt{NaN}\}$ is a strict *weak* order.

Feel free to use this space for scratch work. There's room to write your answer on the next page of this exam.

(Extra space for your answer to Problem Two, Part (iii), if you need it.)

Problem Three: Graph Theory (12 Points)

(Midterm Exam, Fall 2017)

On Problem Set Four, you played around with several different structures that arise in graphs (independent sets, cliques, colorings, and vertex covers). You revisited some of these structures in Problem Set Five. This question explores some additional properties of some of those graph structures.

First, some refreshers from Problem Set Four. An undirected graph $G = (V, E)$ is called ***bipartite*** if there exist two sets V_1 and V_2 such that

- every node $v \in V$ belongs to exactly one of V_1 and V_2, and
- every edge $e \in E$ has one endpoint in V_1 and the other in V_2.

These two sets V_1 and V_2 are called ***bipartite classes*** of G.

An ***independent set*** in a graph $G = (V, E)$ is a set $I \subseteq V$ with the following property:

$$\forall u \in I.\ \forall v \in I.\ \{u, v\} \notin E.$$

A ***vertex cover*** in a graph $G = (V, E)$ is a set $C \subseteq V$ with the following property:

$$\forall u \in V.\ \forall v \in V.\ (\{u, v\} \in E \to u \in C \lor v \in C).$$

Let's now introduce a new definition. We'll say that a ***whimsical set*** in a graph $G = (V, E)$ is a set $W \subseteq V$ such that W is simultaneously an independent set of G and a vertex cover of G.

Prove that if $G = (V, E)$ is a graph and $W \subseteq V$ is a whimsical set in G, then G is a bipartite graph. As a hint, prove that W and $V - W$ are the bipartite classes of G.

Feel free to use this space for scratch work. There's room to write your answer on the next page of this exam.

(Extra space for your answer to Problem Three, if you need it.)

Problem Four: Induction and Cardinality (12 Points)

(Midterm Exam, Fall 2017)

On Problem Set Four, you explored different properties of functions in the context of cardinality. On Problem Set Five, you explored how to combine cardinality arguments and mathematical induction. This problem explores another interplay between the two concepts.

Suppose you have a function $f : A \to A$ from some set to itself where f is injective but ***not*** surjective. It turns out that knowing nothing more than this about the set A, you can conclude that A has to be an infinite set. This question explores why this is.

i. **(1 Point)** Since $f : A \to A$ is not surjective, we know that the following first-order logic statement about f is ***not*** true:

$$\forall x \in A.\ \exists y \in A.\ f(y) = x$$

Since the above formula is ***not*** true, its negation must be true. Negate the above first-order logic statement and simplify it as much as possible.

Negation of this formula: ____

Here's a little scratch space in case you need it.

The negation that you came up with in part (i) of this problem tells us that there's an element $x \in A$ with certain properties. We can use this element x to define the following recurrence relation:

$$\begin{aligned} e_0 &= x, \text{ where } x \text{ is the element of } A \text{ singled out above.} \\ e_{n+1} &= f(e_n) \end{aligned}$$

This recurrence relation has an interesting set of properties.

ii. **(3 Points)** Prove that $e_0 \neq e_1$, that $e_0 \neq e_2$, and that $e_1 \neq e_2$. Remember that $f : A \to A$ is injective but not surjective. It might help to expand out the definitions of e_0, e_1, and e_2 here.

Feel free to use this space for scratch work. There's room to write your answer on the next page of this exam.

(Extra space for your answer to Problem Four, Part (ii), if you need it.)

As a refresher from the previous pages, we're assuming that $f : A \to A$ is a function that is injective but ***not*** surjective. We singled out an element $x \in A$ with some special properties you described in part (i), and used it to define the following recurrence relation:

$$\begin{aligned} e_0 &= x\text{, where } x \text{ is the element of } A \text{ singled out above.} \\ e_{n+1} &= f(e_n) \end{aligned}$$

In part (ii) of this problem, you saw that the first three terms of this sequence are all different. It turns out that this result generalizes to all terms of the sequence.

iii. **(8 Points)** Prove, by induction on n, that this predicate $P(n)$ is true for all $n \in \mathbb{N}$:

$P(n)$ is the statement "for any $m \in \mathbb{N}$ where $m < n$, we have $e_n \neq e_m$."

As a hint on this problem, the reasoning you'll need to use here is very similar to the reasoning for part (ii) of this problem. You might find this problem easier to solve if you explicitly write out what it is that you're assuming at each point in the proof and what it is that you need to prove.

Feel free to use this space for scratch work. There's room to write your answer on the next page of this exam.

(As an aside, what you're proving here shows that no two terms in the sequence are equal. As a result, the set A has to be infinite!)

(Extra space for your answer to Problem Four, Part (iii), if you need it.)