Practice Final Exam VI

We strongly recommend that you work through this exam under realistic conditions rather than just flipping through the problems and seeing what they look like. Setting aside three hours in a quiet space with your notes and making a good honest effort to solve all the problems is one of the single best things you can do to prepare for this exam. It will give you practice working under time pressure and give you an honest sense of where you stand and what you need to get some more practice with.

This practice final exam is essentially the final exam from Fall 2017, with one or two questions swapped out for questions from previous quarter's final exams. The sorts of questions here are representative of what you might expect to get on the upcoming final exam, though the point balance and distribution of problems might be a bit different.

The exam policies are the same for the midterms – closed-book, closed-computer, limited note (one double-sided sheet of 8.5" × 11" paper decorated however you'd like).

You have three hours to complete this exam. There are 70 total points.

Question	Points	Graders
(1) Words of Encouragement	/ 1	
(2) Mathematical Logic and Set Theory	/ 10	
(3) Functions and Binary Relations	/ 16	
(4) Graphs, Pigeonhole, and Induction	/ 16	
(5) Regular and Context-Free Languages	/ 14	
(6) **R** and **RE** Languages	/ 10	
(7) **P** and **NP** Languages	/ 3	
	/ 70	

Problem One: Mathematical Logic and Set Theory (10 Points)

(Final Exam, Fall 2017)

On Problem Set One and Problem Set Two, you explored set theory and mathematical logic. In the course of doing so, you learned how to express yourself precisely using formal mathematical notation, which served as a foundation for the concepts throughout the rest of the quarter. The first two parts of this problem are designed to let you show us what you've learned along the way.

Let's imagine that, in order to foster bipartisanship, Congress passes a law that says that all federal committees must include at least one Republican and at least one Democrat. For the purposes of this problem, you should assume that no one is both a Democrat and a Republican, but that there may be people who are neither Republicans nor Democrats.

i. **(4 Points)** Given the predicates

Committee(x), which says that x is a federal committee;
Member(x, y), which says that x is a member of y;
Democrat(x), which says that x is a Democrat; and
Republican(x), which says that x is a Republican,

write a statement in first-order logic that says "all federal committees have at least one Republican member and at least one Democratic member."

In a mathematical sense, we can think of a committee as a set whose elements are the people on that committee. This lets us talk about different committees in the language of set theory.

ii. **(3 Points)** Let S be the set of all people in the United States, R be the set of all Republicans, and D be the set of all Democrats. Using set theory notation (e.g. $\cup$, $\subseteq$, $\wp$, $\in$, etc.), but ***without*** using set-builder notation and ***without*** using first-order logic, write an expression that represents the set of all possible committees of people from the US that include at least one Democrat and at least one Republican.

    As before, remember that some people may be neither Republicans nor Democrats.

(The last part of this problem has nothing to do with the first two parts. ☺)

On Problem Set Two, you explored how it's possible to express every formula in propositional logic using only a strict subset of the propositional connectives. In particular, you showed that every propositional formula can be rewritten purely using the $\rightarrow$ and $\bot$ connectives.

It turns out that every propositional formula can also be rewritten purely using the $\vee$, $\leftrightarrow$, and $\bot$ connectives. One way to show this is to express the $\rightarrow$ connective just using $\vee$, $\leftrightarrow$, and $\bot$.

iii. **(3 Points)** Without using any propositional connectives besides $\vee$, $\leftrightarrow$, and $\bot$, write a formula in propositional logic that's equivalent to $p \rightarrow q$.

Problem Two: Functions and Binary Relations (16 Points)

(Final Exam, Fall 2017)

On Problem Sets Three, Four, and Five, you explored different properties of strict orders. Although strict orders come in all sorts of shapes and flavors, there is a single strict order that's, in some sense, the "most fundamental" strict order: the strict subset relation. In this problem, you'll show that every strict order's behavior can be thought of as the behavior of the strict subset relation over some well-chosen collection of sets.

Let R be a strict order over a set A. Consider the function $f : A \to \wp(A)$ defined as follows:

$$f(x) = \{\ y \in A \ \mid \ y = x \text{ or } yRx\ \}$$

This function f connects the relation R over A to the relation $\subsetneq$ over $\wp(A)$.

i. **(6 Points)** Prove for all a, $b \in A$ that if $f(a) \subsetneq f(b)$, then aRb. As a reminder, the notation $S \subsetneq T$ means that $S \subseteq T$ and that $S \neq T$.

   Feel free to use the space below for scratch work. There's room for your answer to this question 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, we've let R be a ***strict order*** over a set A and defined the function $f : A \to \wp(A)$ as follows:

$$f(x) = \{\ y \in A \ \mid \ y = x \ \text{ or } \ yRx\ \}$$

ii. **(10 Points)** Prove for all $a, b \in A$ that if aRb, then $f(a) \subsetneq f(b)$. Again, the notation $S \subsetneq T$ means that $S \subseteq T$ and that $S \neq T$.

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

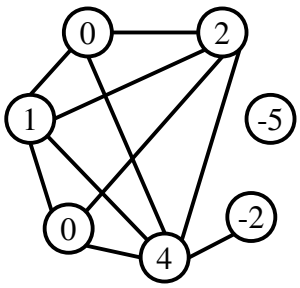
(Extra space for your answer to Problem Two, Part (ii), if you need it.)

Problem Three: Graphs, Pigeonhole, and Induction (16 Points)

(Final Exam, Fall 2017)

On Problem Set Four and Problem Set Five, you explored different classes of graphs and their properties. This problem concerns a new family of graphs and how those graphs connect to other concepts and techniques you've seen this quarter.

Let's begin with a new definition. A ***threshold graph*** is a graph $G = (V, E)$ where each node $v \in V$ is assigned an integer called its ***weight***, denoted $s(v)$, and there's an edge between any two distinct nodes u and v if and only if $s(u) + s(v) > 0$. There's a sample threshold graph shown to the right. Notice, for example, that there's an edge between the node labeled 4 and the node labeled -2 because $4 + (-2) > 0$. However, there's no edge between the node labeled -2 and the node labeled 2, since $2 + (-2) \not> 0$.



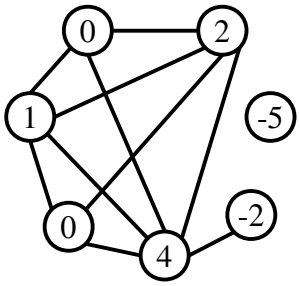
i. **(6 Points)** Prove that if $G = (V, E)$ is a threshold graph and $|V| = 2n+1$ for some natural number n, then either G contains a clique of size at least $n+1$ or G contains an independent set of size at least $n+1$ (or both).

As a hint, every integer either is positive or is not positive.

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

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

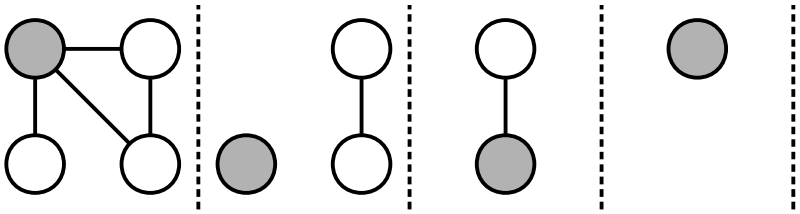
As a refresher, a ***threshold graph*** is a graph $G = (V, E)$ where each node $v \in V$ is assigned an integer called its ***weight***, denoted $s(v)$, and there's an edge between any two distinct nodes u and v if and only if $s(u) + s(v) > 0$.



Now, a new definition. A graph G is called ***totally reducible*** if either

- G has no nodes, or
- G has a node that is either adjacent to *every other* node in G or adjacent to *no other* nodes in G, and deleting that node leaves behind a totally reducible graph.

Intuitively, a totally reducible graph is one that can be converted to an empty graph by repeatedly selecting and deleting some node that's either adjacent to each other node in the graph or adjacent to no other node in the graph. For example, the graph shown below to the left is totally reducible, which you can see by deleting the highlighted node at each step:



It turns out that every threshold graph is totally reducible. You can see this in the sample threshold graph shown above by deleting the nodes in the order -5, 4, -2, 2, 1, 0, 0.

ii. **(10 Points)** Prove by induction that if G is a threshold graph, then G is totally reducible.

As a hint, focus on two nodes v_{max} and v_{min} whose weights are the highest and lowest in the graph and look at $s(v_{max}) + s(v_{min})$. If v_{max} and v_{min} have the highest and lowest weights of any of the nodes in the graph, then $s(u) \leq s(v_{max})$ and $s(u) \geq s(v_{min})$ for any $u \in V$ (you don't need to prove this.)

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

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

Problem Four: Regular and Context-Free Languages (14 Points)

(Final Exam, Fall 2017)

Let $\Sigma = \{$ `h`, `i`, `m`, `r`, `t` $\}$ and consider the following language L_1:

$$L_1 = \{\ w \in \Sigma^* \mid w \text{ is a substring of } \texttt{mirth}\ \}.$$

Recall that a substring is a contiguous range of characters taken out of an original string. For example, `mir` $\in L_1$, `irt` $\in L_1$, $\varepsilon \in L_1$, `t` $\in L_1$, and `mirth` $\in L_1$, but `mrh` $\notin L_1$ (although the letters in `mrh` appear in `mirth`, they're not contiguous), `it` $\notin L_1$ (for the same reason `mrh` is not in L_1), and `mmm` $\notin L_1$ (because there aren't three consecutive `m`'s in `mirth`).

i. **(3 Points)** Design an NFA for L_1. In the space at the bottom of the page, write a brief explanation (at most two sentences) for how your NFA works.

Explanation for this NFA (at most two sentences):

On Problem Sets Six, Seven, and Eight, you explored languages involving taking a walk with your dog. The next two parts of this problem concern more of the challenges of pet ownership.

Imagine that you and your dog are taking a walk and you have a leash that's six units long. Unlike before, you and your dog don't move at the same speed. Every time your dog takes a step, your dog moves three units forward, and every time you take a step, you move two units forward.

Let $\Sigma = \{\texttt{y}, \texttt{d}\}$, where `y` represents you taking a step (which moves you two units forward) and `d` represents your dog taking a step (which moves your dog three units forward) and consider the following language:

$$L_2 = \{\ w \in \Sigma^* \mid w \text{ represents a walk where you and your dog end at the same position, you and your dog are never more than six units apart, and you never are ahead of your dog.}\ \}$$

For example, the string `ddyyy` $\in L_2$, as are `dydydydyyy`, ε, `dydyyddyyy`, and `dydydyydyy`. However, the string `yddyy` $\notin L_2$ (since after the first step you end up ahead of your dog), the string `ddyddyyyyy` $\notin L_2$ (at the underlined point, your dog is more than six units ahead of you), and the string `dy` $\notin L_2$ (your dog ends up one unit ahead of you).

ii. **(4 Points)** Design a DFA for L_2. At the bottom of the page, write a very brief explanation (at most two sentences) about how your DFA works.

Explanation for this DFA (at most two sentences):

Now, let's imagine that you're taking your dog for a walk but you take off the leash. As before, every time your dog takes a step it moves three units forward, and every time you take a step you move two units forward.

Let $\Sigma = \{\texttt{y}, \texttt{d}\}$, where `y` represents you taking a step (which moves you two units forward) and `d` represents your dog taking a step (which moves your dog three units forward) and consider the following language L_3:

$$L_3 = \{\ w \in \Sigma^* \ \mid\ w \text{ represents a walk where you and your dog end at the same position and you never are ahead of your dog. } \}$$

This is essentially the same language as L_2, except without the leash restriction. This language is not regular, and in this problem we'd like you to convince us why this is.

iii. **(3 Points)** To save you time, rather than having you write out a full proof that L_3 is not regular, we'd instead like you to answer the following questions:

1. The Myhill-Nerode theorem requires you to choose an infinite set S of strings that are pairwise distinguishable relative to L_3. In the space below, write such a set S.

2. Suppose you choose two arbitrary, distinct strings x and z from the set S. What string w will you append to x and z to show that they are distinguishable relative to L_3?

3. Briefly explain why the choice of w you found above shows that x and z are distinguishable relative to L_3.

Let Σ = { `3`, `8`, `+`, `(`, `)` } and consider the following language L_4:

L_4 = { $w \in \Sigma^*$ | w is a syntactically correct mathematical expression for an even number }

For example, the following strings are in L_4:

- `8`
- `38`
- `8 + 8`
- `3 + 3`
- `33 + 38 + 83 + 88`
- `(8)`
- `(8 + 3 + 3 + (3 + 3))`
- `((((88333388))))`

The following strings are *not* in L_4. Note in particular that we do ***not*** allow for leading + signs on numbers and that we do ***not*** allow for implicit multiplication (we're just allowing addition):

- ε
- `3`
- `8 + 8 + 3`
- `33388833`
- `(((8+3)`
- `++3` *(math isn't C++. ☺)*
- `+8` *(no leading + signs)*
- `8(3)` *(no implicit multiplication)*

This language happens to be context-free.

iv. **(4 Points)** Write a CFG for L_4. In the space at the bottom of the page, write a brief explanation (at most two sentences) for how your CFG works.

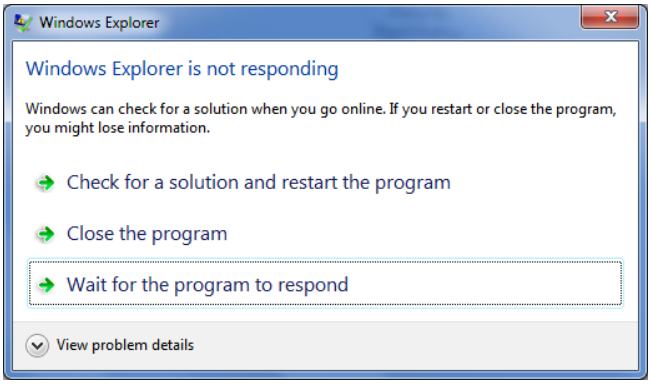
As a hint, you'll almost certainly want to use multiple nonterminals.

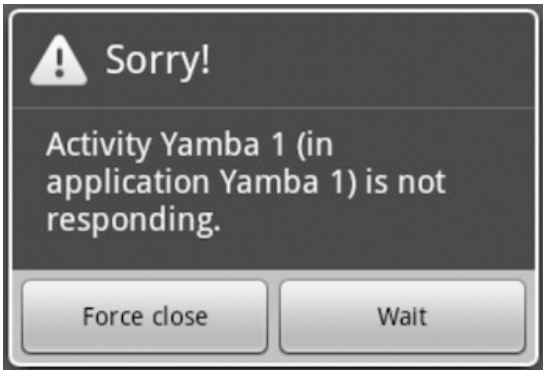
Explanation for this CFG (at most two sentences):

Problem Five: R and RE Languages (10 Points)

(Final Exam, Fall 2011)

Most operating systems provide some functionality to detect programs that are looping infinitely. Typically, they display a dialog box containing a message like these:



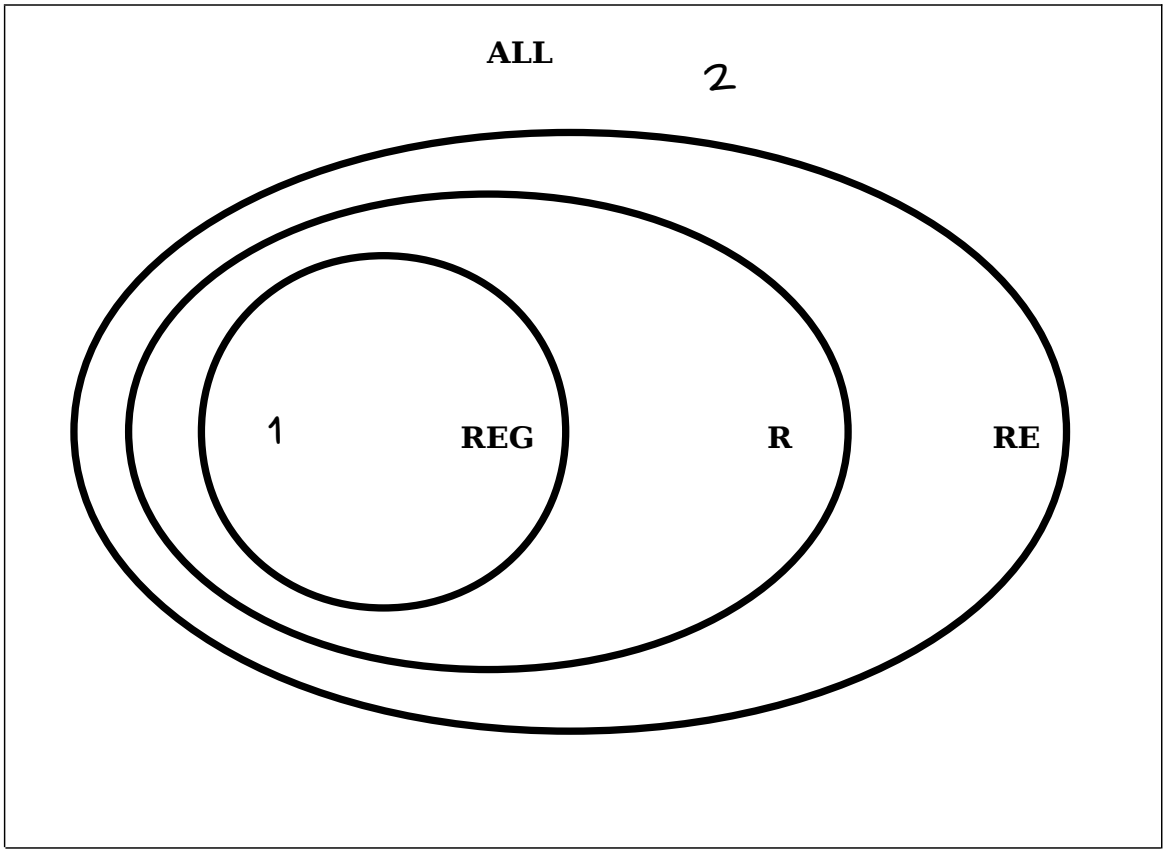


These messages give the user the option to terminate the program or to let the program keep running.

An ideal OS would shut down any program that had gone into an infinite loop, since these programs just waste system resources (processor time, battery power, etc.) that could be better spent by other programs.

i. **(3 Points)** Since it makes more sense for the OS to automatically detect programs that have gone into an infinite loop, why does the OS have to ask the user whether to terminate the program or let it keep running?

ii. **(7 Points)** Below is a Venn diagram showing the overlap of different classes of languages we've studied so far. We have also provided you a list of numbered languages. For each of those languages, draw where in the Venn diagram that language belongs. As an example, we've indicated where Language 1 and Language 2 should go. No proofs or justifications are necessary, and there is no penalty for an incorrect guess.



1. Σ^*
2. L_D
3. $\{ w \in \{\texttt{a}, \texttt{b}\}^* \mid w$ is ***not*** a palindrome $\}$
4. $\{ wxyxz \mid w, x, y, z \in \{\texttt{a}, \texttt{b}\}^*$ and $|x| = 5 \}$
5. $\{ w \in \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{r}\}^* \mid w$ is ***not*** a substring of `abracadabra` $\}$
6. $\{ w \in \{\texttt{a}, \texttt{b}\}^* \mid$ there is a TM M where M loops on $w \}$
7. $\{ \langle M \rangle \mid M$ is a TM that accepts $\langle 137 \rangle$ and rejects $\langle 42 \rangle \}$
8. $\{ \langle M \rangle \mid M$ is a TM that does not accept $\langle 137 \rangle$ or does not reject $\langle 42 \rangle \}$
9. $\{ \langle M, n \rangle \mid M$ is a TM, $n \in \mathbb{N}$, and M accepts at least one string of length $n \}$

Problem Six: P and NP Languages (3 Points)

(Final Exam, Fall 2017)

Here's a quick series of true/false questions about the **P** and **NP** languages. Each correct answer is worth one point, and there is no penalty for an incorrect guess. You do not need to justify your answers.

i. If M is a decider for a language L and $L \in \mathbf{P}$, then M runs in polynomial time.

☐ True ☐ False

ii. If V is a polynomial-time verifier for a language L, then $V \in \mathbf{NP}$.

☐ True ☐ False

iii. If $L \in \mathbf{RE}$, then $L \in \mathbf{NP}$.

☐ True ☐ False

We have one final question for you: do *you* think $\mathbf{P} = \mathbf{NP}$? Let us know in the space below. There are no right or wrong answers to this question – we're honestly curious to hear your opinion!

☐ I think $\mathbf{P} = \mathbf{NP}$ ☐ I think $\mathbf{P} \neq \mathbf{NP}$