CS103 Final Exam

This exam is open-book and open-note. You may use a computer only to look at notes that you yourself have written, to access the course website and the tools there, and to read an online copy of one of the recommended readings. No other use of the computer is permitted during this exam. You must hand-write all of your solutions on this physical copy of the exam. No electronic submissions will be considered without prior consent of the course staff.

SUNetID: ____________________

Last Name: ____________________

First Name: ____________________

I accept both the letter and the spirit of the honor code. I have not received any unpermitted assistance on this test, nor will I give any. My answers are my own work and are not adapted from any unpermitted sources. I will not use a computer except in the ways specified at the top of the exam.

(signed) ____

You have three hours to complete this exam. There are 180 total points, and this exam is worth 25% of your total grade in this course. You may find it useful to read through all the questions to get a sense of what this exam contains. As a rough sense of the difficulty of each question, there is one point on this exam per minute of testing time.

Question		**Points**	**Grader**
(1) Discrete Mathematics	(20)	/ 20	
(2) Regular Languages	(45)	/ 45	
(3) Context-Free Languages	(20)	/ 20	
(4) **R**, **RE**, and co-**RE** Languages	(45)	/ 45	
(5) **P** and **NP** Languages	(20)	/ 20	
(6) The Big Picture	(30)	/ 30	
	(180)	**/ 180**	

It has been a pleasure teaching CS103 this quarter. Good luck on the final exam!

Problem One: Discrete Mathematics (20 Points Total)

A doctor has prescribed a patient medicine that is absorbed into the bloodstream. The medicine has a *half-life* of one hour, meaning that each hour, half of the medicine in the patient's bloodstream will be removed by her body. For example, if the patient had 5mg of the medicine in her bloodstream at 6:00PM, then at 7:00PM she would have 2.5mg of the medicine in her bloodstream.

Suppose that the doctor gives the patient 1mg of the medicine at the start of every hour, all of which is immediately absorbed into her bloodstream. You are concerned because each time the patient receives a dose, some amount of the medicine will still be left in her bloodstream. Wouldn't this give the patient a dangerous amount of medicine?

Fortunately, now that you've taken a course in discrete math, you can determine exactly how much medicine will be in the patient's bloodstream, which will help you determine whether she will ever have a dangerous amount of the medicine in her blood.

Let c_n denote the amount of active medication in the patient's body n hours after the first dose has been administered. The first dose is 1mg, so c_0 = 1mg. One hour later, half of the medicine will have been cleared from her bloodstream, leaving 0.5mg, and the patient will receive 1mg more medicine, bringing the total up to 1.5mg. Thus c_1 = 1.5mg. An hour after that, half that medicine will have been cleared from her bloodstream, leaving 0.75mg of medicine in her bloodstream, and the patient will then receive another 1mg of medicine, bringing the total up to 1.75mg. Thus c_2 = 1.75mg.

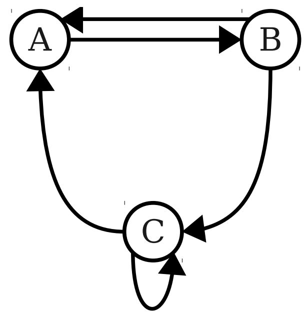
Prove, by induction, that $c_n = (2 - 1/2^n)$mg for all $n \in \mathbb{N}$. This proves that the patient will never have more than 2mg of medicine in her bloodstream, even if she continues to take 1mg doses every hour.

(Extra space for Problem One, if you need it)

Problem Two: Regular Languages (45 Points Total)

Recall that a *path* in a graph is a series of nodes $v_1, v_2, \ldots, v_n$ such that each pair of adjacent nodes in the path is connected by an edge.

Consider the following graph G:



Let $\Sigma = \{$**A**, **B**, **C**$\}$. We can represent a path in G as a nonempty string where the letters spell out the path in the graph. For example, the path **A**, **B**, **C**, **C** would be represented by the string **ABCC**.

(i) Path Automata (10 Points)

Let $L = \{ w \in \Sigma^* \mid w \text{ represents a path in } G \}$, where G is the graph given above. For example:

A $\in L$	**ACC** $\notin L$
ABC $\in L$	$\varepsilon \notin L$
BCC $\in L$	**BBA** $\notin L$
CCABA $\in L$	**ABBC** $\notin L$

Design a DFA for L.

(ii) Regular Expressions for Inequalities **(10 Points)**

Let $\Sigma = \{$**1**, **2**, **≤**$\}$ and let L be the language defined as follows:

$$L = \{\ w \in \Sigma^* \mid w \text{ is a valid chain of inequalities relating the numbers } \mathbf{1} \text{ and } \mathbf{2}\ \}.$$

For example:

1≤2 $\in L$	**2≤≤** $\notin L$
1≤1≤2≤2 $\in L$	**≤2** $\notin L$
2≤2≤2 $\in L$	$\varepsilon \notin L$
1≤1≤1≤1 $\in L$	**1** $\notin L$
1≤1≤2 $\in L$	**12≤22** $\notin L$

Note in particular that inequalities involving numbers like 12, 222, 121212, etc. whose digits are 1 and 2 aren't allowed (the inequality should only relate the numbers 1 and 2) and any individual number itself isn't allowed.

Write a regular expression for L.

(iii) Distinguishability **(5 Points)**

Let L be an arbitrary language. Prove that if $x \in L$ and $y \notin L$, then x and y are distinguishable relative to L.

(iv) State Lower Bounds **(10 Points)**

In Problem Set 6, you proved that any DFA for the language

$$L = \{ w \in \{\mathtt{0}, \mathtt{1}\}^* \mid w \text{ contains at least two } \mathtt{1}\text{'s separated by exactly 5 characters} \}$$

must have at least 64 states. One possible proof of this result (which is given in our solution set) is to consider the set of all strings of `0`s and `1`s whose length is exactly six. There are 64 of these strings, and all of them are distinguishable relative to L.

It turns out you can prove a stronger claim: any DFA for this language must have at least 65 states. Using your result from part (iii), prove that any DFA for L must have at least 65 states. *(Hint: The above logic gives you a set of 64 strings distinguishable relative to L and you can use this fact without proof. Can you find a string distinguishable from all of them?)*

(v) State Lower Bounds, Part Two **(10 Points)**

In Problem Set Five, you designed an NFA for the language

$$L = \{ w \in \{\mathtt{0}, \mathtt{1}\}^* \mid w \text{ contains at least two } \mathtt{1}\text{'s separated by exactly 5 characters} \}$$

Using your result from part (iv), which says that any DFA for L must have at least 65 states, prove that every NFA for L must have at least seven states. *(Hint: Think about the subset construction.)*

Problem Three: Context-Free Languages (20 Points)

In Problem Set Six, you designed a CFG for the following language:

$$ADD = \{\ \mathtt{1}^m\mathtt{+1}^n\overset{?}{=}\mathtt{1}^{m+n} \mid m, n \in \mathbb{N}\ \}$$

Now, consider the following language over the alphabet {1, +, ≈}, which is a variation on *ADD*:

$$NEAR = \{\ \mathtt{1}^m\mathtt{+1}^n\mathtt{\approx}1^p \mid m, n, p \in \mathbb{N} \text{ and } m + n = p + 1\ \}$$

Intuitively, *NEAR* is the set of all arithmetic expressions where the left-hand side is exactly one greater than the right-hand side. For example:

111+1≈111 ∈ *NEAR* | **+≈** ∉ *NEAR*

11+111≈1111 ∈ *NEAR* | **1+1≈11** ∉ *NEAR*

1+≈ ∈ *NEAR* | **1+1≈111** ∉ *NEAR*

+1≈ ∈ *NEAR* | **1+1+1≈11** ∉ *NEAR*

Write a CFG for *NEAR*. Then, draw parse trees for **11+1≈11** and **1+11≈11**. We've included space to draw those parse trees at the bottom of the page.

1	1	+	1	≈	1	1

1	+	1	1	≈	1	1

Problem Four: R, RE, and co-RE Languages (45 Points Total)

A *palindrome* is a string that is the same forwards and backwards. Consider the following language L:

$$L = \{ \langle M \rangle \mid M \text{ is a TM and every string in } \mathscr{L}(M) \text{ is a palindrome} \}$$

(i) Palindromes and RE Languages (25 Points)

Prove that $L \notin \mathbf{RE}$.

(Extra space for Problem 4.i, if you need it)

$$L = \{ \langle M \rangle \mid M \text{ is a TM and every string in } \mathscr{L}(M) \text{ is a palindrome } \}$$

(ii) Palindromes and co-RE Languages **(20 Points)**

Prove that $L \in$ co-**RE**.

(Extra space for Problem 4.ii, if you need it)

Problem Five: P and NP Languages (20 Points)

Recall from lecture that the language *3COLOR* = { ⟨*G*⟩ | *G* is a 3-colorable graph } is **NP**-complete. The language *2COLOR* = { ⟨*G*⟩ | *G* is a 2-colorable graph } is known to be in **P** (you don't need to prove this). Below is a purported proof that **P** = **NP**:

Theorem: **P** = **NP**.

Proof: As we will prove in the lemma below, we have *2COLOR* $\le_P$ *3COLOR*. Since *3COLOR* is **NP**-hard, this means *2COLOR* is **NP**-hard. Because *2COLOR* ∈ **P** and **P** ⊆ **NP**, we know that *2COLOR* ∈ **NP**. Thus *2COLOR* ∈ **NP** and *2COLOR* is **NP**-hard, so **NP**-complete. Since *2COLOR* is **NP**-complete and *2COLOR* ∈ **P**, we thus have that **P** = **NP**. ■

Lemma: 2COLOR $\le_P$ *3COLOR*.

Proof: We'll give a polynomial-time mapping reduction from *2COLOR* to *3COLOR*, which proves that *2COLOR* $\le_P$ *3COLOR*.

Given a graph $G = (V, E)$, let $f(\langle G \rangle)$ be the graph G' defined in terms of G by adding a new node v to G and adding an edge from v to each other node in G. We state without proof that f can be computed in polynomial time. Therefore, we will prove that $\langle G \rangle \in$ *2COLOR* iff $f(\langle G \rangle) = \langle G' \rangle \in$ *3COLOR*, from which we can conclude that f is a polynomial-time mapping reduction from *2COLOR* to *3COLOR*, so *2COLOR* $\le_P$ *3COLOR*.

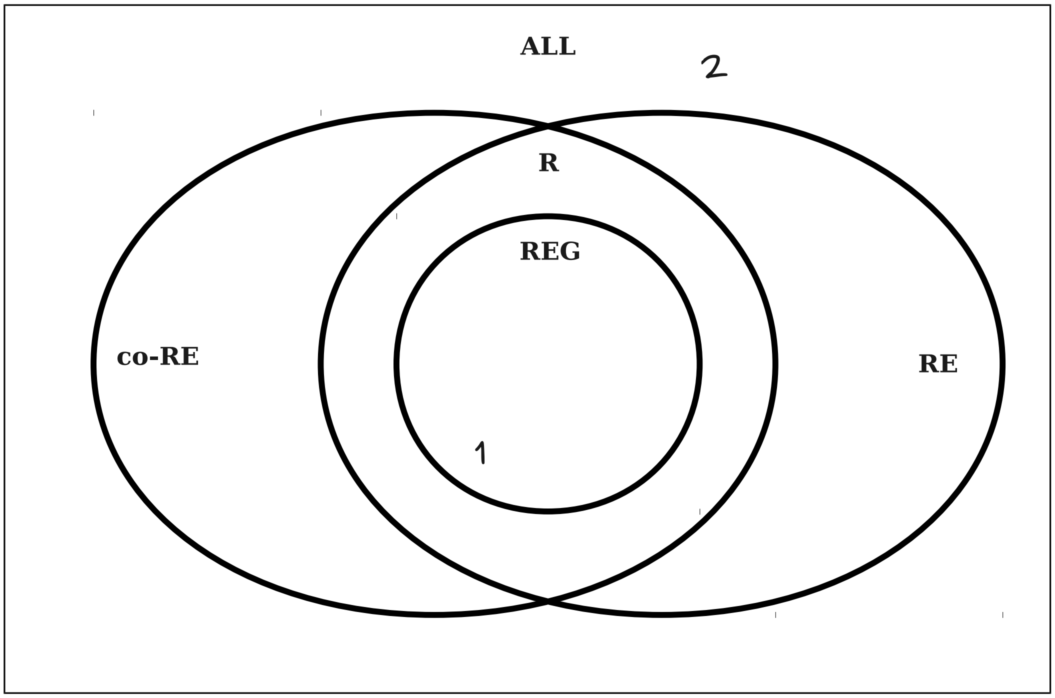
First, we prove that if G is 2-colorable, then G' is 3-colorable. To see this, consider any 2-coloring of G. For each node in G' that also appears in G, color that node the same color as its corresponding node in G. Then, color v the unused third color. This gives a 3-coloring of G'.

Next, we prove that if G' is 3-colorable, then G is 2-colorable. To see this, consider any 3-coloring of G'. We claim that all of the nodes in G' that also belong to G are colored with only two colors. To see this, note that since all these nodes are connected to the new node v, they must all have a color distinct from v's color, and thus must be colored using only two colors. Therefore, if we color the nodes in G the same color as the corresponding nodes in G', we end with a 2-coloring of G. ■

The above proof, unfortunately, is incorrect. What is wrong with this proof? Be specific. (The function f described in the lemma can indeed be computed in polynomial time, so that isn't the error in the proof.)

Problem Six: The Big Picture (30 Points Total)

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 12 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.



1. Σ^*
2. EQ_{TM}
3. $\{ \mathtt{a}^n \mid n \in \mathbb{N} \}$
4. $\{ \mathtt{a}^n \mid n \in \mathbb{N}$ and is a multiple of 137. $\}$
5. $\{ \mathtt{a}^n \mid n \in \mathbb{N}$ and is not prime. $\}$
6. $\{ \langle M \rangle \mid M$ is a Turing machine and $\mathscr{L}(M)$ is finite. $\}$
7. $\{ \langle M \rangle \mid M$ is a Turing machine and $\mathscr{L}(M) = L_D$. $\}$
8. $\{ \langle M \rangle \mid M$ is a Turing machine and $\mathscr{L}(M) = A_{TM}$. $\}$
9. $\{ \langle M, n \rangle \mid M$ is a TM, $n \in \mathbb{N}$, and M ***accepts*** all strings of length at most n. $\}$
10. $\{ \langle M, n \rangle \mid M$ is a TM, $n \in \mathbb{N}$, and M ***rejects*** all strings of length at most n. $\}$
11. $\{ \langle M, n \rangle \mid M$ is a TM, $n \in \mathbb{N}$, and M ***loops*** on all strings of length at most n. $\}$
12. $\{ \langle M_1, M_2, M_3, w \rangle \mid M_1, M_2$, and M_3 are TMs, w is a string, and at least two of M_1, M_2, and M_3 accept w. $\}$