CS103 Final Exam

This final exam is open-book, open-note, open-computer, but closed-network. This means that if you want to have your laptop with you when you take the exam, that's perfectly fine, but you **must not** use a network connection. You should only use your computer to look at notes you've downloaded in advance. Although you may use computers during the 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 assistance on this test, nor will I give any.

(signed) ______________________________________

You have three hours to complete this final exam. There are 180 total points and 180 minutes, so you may find it useful to use the point totals as a rough estimate about how to spend your time. This exam is worth 20% 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.

Good luck!

Question		**Points**	**Grader**
(1) Mathematical Languages	(20)	/ 20	
(2) Regular Languages	(35)	/ 35	
(3) Context-Free Languages	(30)	/ 30	
(4) Recursive and RE Languages	(55)	/ 55	
(5) P and NP Languages	(40)	/ 40	
	(180)	**/ 180**	

Problem One: Mathematical Languages (20 points)

In each of the following, you will be given a list of first-order predicates and functions along with an English sentence. In each case, write a statement in first-order logic that expresses the indicated sentence. The statement you write can use any first-order construct (equality, connectives, quantifiers, etc.), but you must only use the predicates and functions provided.

As an example, if you were given the predicate *Integer(x)*, which returns whether x is an integer, and the function *Plus(x, y)*, which returns $x + y$, you could write the statement "there is some even integer" as

$$\exists n.\ \exists k.\ (Integer(n) \land Integer(k) \land Plus(k, k) = n)$$

since this asserts that some number n is equal to $2k$ for integer k. However, you could not write

$$\exists n.\ (Integer(n) \land Even(n))$$

because there is no *Even* predicate.

(i) The Most Interesting Meme on the Final (10 Points Total)

Given the predicates

Time(t), which states that *t* is a time, and
WritesLogic(p, t), which means that *p* writes logic during *t*,

along with the constants

I, which refers to you, and
CS103FinalTime, which refers to the time this exam is given,

write a statement in first-order logic that says "I don't always write logic, but when I do, it's during the CS103 final exam."

(ii) Logic Meets DFAs **(10 Points Total)**

Given the predicates

> *Regular(L)*, which states whether *L* is a regular language,
> *String(w)*, which states whether *w* is a string,
> *Contains(L, w)*, which states whether *L* contains *w*,
> *DFA(D)*, which states whether *D* is a DFA, and
> *Accepts(D, w)*, which states whether *D* accepts *w*,

write a statement in first-order logic that says "A language is regular if and only if there is some DFA that accepts precisely the strings in that language."

Problem 2: Regular Languages (35 points total)

Consider the following language over $\Sigma = \{ \texttt{0}, \texttt{1} \}$:

$$SANDWICH = \{ w \mid w \text{ starts and ends with the same symbol, and } |w| > 0 \}$$

For example, `010` ∈ *SANDWICH,* `111` ∈ *SANDWICH,* `1` ∈ *SANDWICH,* and `00` ∈ *SANDWICH.* However, ε ∉ *SANDWICH* (because it has length zero), `10` ∉ *SANDWICH* (because it doesn't begin and end with the same symbol), and `0101` ∉ *SANDWICH.*

(i) Finite Automata (10 Points)

Design a DFA that accepts *SANDWICH.*

(ii) Regular Expressions (10 Points)

Write a regular expression for *SANDWICH.*

Consider the following language over the alphabet Σ = {`0`, `1`, `?`}:

$$CONTAINS = \{\ t\texttt{?}p \mid t, p \in \{\texttt{0}, \texttt{1}\}^* \text{ and } p \text{ is a substring of } t\ \}$$

This language is similar to the *CONTAINS* language from Problem Set 6 and Problem Set 7, except that the ordering of the text and pattern string is reversed such that the text appears before the pattern. For example, `0011?01` ∈ *CONTAINS* because `01` appears in `0011`, `1100?110` ∈ *CONTAINS* because `110` is a substring of `1100`, and `100?` ∈ *CONTAINS* because the empty string is a substring of `100`. However, `10?1100` ∉ *CONTAINS* because `1100` is not a substring of `10`, `111000?01` ∉ *CONTAINS* because `01` is not a substring of `111000`, and `10101?100` ∉ *CONTAINS* because `100` is not a substring of `10101`.

(iii) The Pumping Lemma **(15 Points)**

Using the pumping lemma for regular languages, prove that *CONTAINS* is not regular. To help out, we have sketched out a part of the proof; you should fill in the appropriate blanks.

Proof: By contradiction; assume that *CONTAINS* is regular. Let n be the length guaranteed by the pumping lemma. Then consider the string w = $\boxed{}$
We then have that $w \in CONTAINS$ and $|w| \geq n$, so by the pumping lemma we can write $w = xyz$ such that $|xy| \leq n$, $y \neq \varepsilon$, and for any i ∈ ℕ, $xy^iz \in CONTAINS$.

(finish the proof in the box below)

We have reached a contradiction, so our assumption was wrong and *CONTAINS* is not regular. ■

Problem 3: Context-Free Languages (30 points total)

Consider the following language over the alphabet $\Sigma = \{\texttt{0}, \texttt{1}\}$:

$$LE = \{ \texttt{0}^m \texttt{1}^n \mid m, n \in \mathbb{N} \text{ and } m \leq n \}$$

Informally, *LE* is the language of strings of some number of **0**s followed by at least as many **1**s. For example, **011** ∈ *LE*, **11** ∈ *LE*, ε ∈ *LE*, but **00011** ∉ *LE* and **01110** ∉ *LE*.

(i) The First Step is Admitting It (5 Points)

Below is a grammar for *LE*:

$$S \rightarrow \texttt{0}S\texttt{1} \mid S\texttt{1} \mid \varepsilon$$

The above grammar is ambiguous. Show that it is ambiguous by finding a string in *LE* with at least two different parse trees and draw two of those parse trees.

(ii) Repairing the Grammar (10 Points)

Design an unambiguous grammar for *LE*.

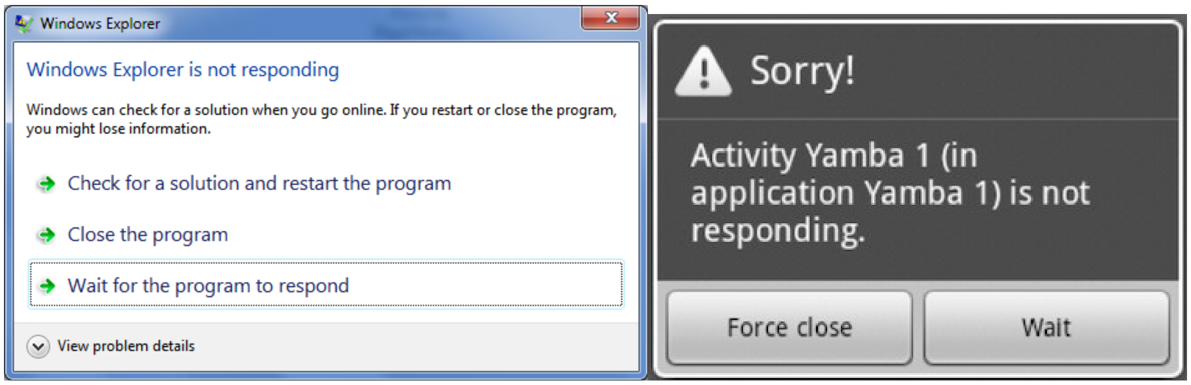
(iii) Designing DPDAs **(15 Points)**

Design a **deterministic** PDA that recognizes *LE*. Recall that a PDA is deterministic if for each state/input/stack combination, there is at most one transition that can be followed at any time.

Problem 4: R and RE Languages (55 points total)

(i) This Program is Not Responding (5 Points)

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



Windows Explorer is not responding

Windows can check for a solution when you go online. If you restart or close the program, you might lose information.

Sorry!

Activity Yamba 1 (in application Yamba 1) is not responding.

Windows Explorer
Check for a solution and restart the program
Close the program
Wait for the program to respond
View problem details
Force close
Wait

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. 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) Rice's Theorem **(15 Points)**

Below are five languages. For each language, check the YES column if Rice's theorem applies to the language and NO if it does not. Note that you are **not** checking YES or NO based on whether or not the language is decidable; just answer whether Rice's theorem applies. You do not need to prove why your answers are correct.

	YES	NO
L = { ⟨M⟩ | L(M) ∈ NP }	_______	_______
L = { ⟨M⟩ | M rejects exactly five strings }	_______	_______
L = { ⟨M⟩ | L(M) is regular but not recursive }	_______	_______
L = { ⟨M, *n*⟩ | L(M) contains exactly *n* strings }	_______	_______
L = { ⟨M⟩ | There is a DFA D such that D accepts *w* iff M accepts *w* }	_______	_______

(iii) R, RE, and Non-RE Languages **(10 Points)**

Let L be any language. Prove that if L is undecidable and $\overline{\text{L}}$ is RE, then L is **not** RE.

(iv) TMs and Regular Languages **(25 Points)**

Consider the following language:

$$\text{EQ}_{\text{TM/REG}} = \{ \langle M, R \rangle \mid M \text{ is a TM, } R \text{ is a regular expression, and } L(R) = L(M) \}$$

For example, if M is a TM that accepts just the strings **`puppy`** and **`kitty`** and R is the regular expression **`puppy|kitty`**, then $\langle M, R \rangle \in \text{EQ}_{\text{TM/REG}}$.

Prove that $\text{EQ}_{\text{TM/REG}}$ is undecidable by reducing *HALT* to it.

Problem 5: P and NP Languages (40 points total)

(i) NP-Completeness? (5 Points)

Suppose that you find a polynomial-time reduction from some language L to 3SAT. Why can't you conclude that L is NP-complete?

(ii) NP-Complete Problems are Hard (15 Points)

Suppose that there is a language $L \in NP$ that cannot be decided in polynomial time (that is, $L \notin P$). Prove that in this case, no NP-complete language can be decided in polynomial time.

(iii) P vs NP **(20 Points)**

This problem explores the question

What would it take to prove whether or not P = NP?

Below are ten statements. For each statement, if the statement would definitely prove that P = NP, write "P = NP" on the appropriate line. If the statement would definitely prove that P ≠ NP, write "P ≠ NP" on the appropriate line. If the statement would not prove either result, write "neither" on the appropriate line.

There is an NP language that can be decided in polynomial time. _________

There is an NP-**complete** language that can be decided in polynomial time. _________

There is an NP-**hard** language that can be decided in polynomial time. _________

There is an NP language that cannot be decided in polynomial time. _________

There is an NP-**complete** language that cannot be decided in polynomial time. _________

There is an NP-**hard** language that cannot be decided in polynomial time. _________

There is **some** NP-complete language that can be decided in $O(2^n)$ time. _________

There is **no** NP-complete language that can be decided in $O(2^n)$ time. _________

There is a polynomial-time verifier for every problem in NP. _________

There is a polynomial-time 137-approximation to TSP. _________