Extra Credit: Practice CS103 Final Exam

This practice exam is worth **5 extra credit points**. We will not give points based on whether or not your answers are correct, but rather on whether or not you have made a good-faith effort to answer all the questions. On the Honor Code, we assume that any answers you submit for these problems represent a good, honest effort on your part. We will **not** release solutions to this practice exam. If you have any questions about it, feel free to stop by office hours, email the staff list, or stop by one of the final exam review sessions we'll be holding. It is perfectly fine to work on these problems in a group or to ask questions about them at the review session.

Normally, I would leave extra space between problems so that you would have room to write out your answers, but to save paper I have tried to minimize the amount of blank space in this handout. You do not need to bring extra scratch paper to the final exam, but I would suggest doing so in case you want to try out various solutions to the problems. You will have three hours to complete the final exam.

This practice exam is probably a bit larger than the actual exam will be, so don't worry if you take more than three hours to finish it. We will release two actual exams from previous quarters early next week so that you can get a sense for what the final exam might be like.

Question		**Points**	**Grader**
(1) Discrete Mathematics	(25)	/ 25	
(2) Regular Languages	(45)	/ 45	
(3) Context-Free Languages	(15)	/ 15	
(4) **R** and **RE** Languages	(55)	/ 55	
(5) **P** and **NP** Languages	(40)	/ 40	
	(180)	**/180**	

(Note that these points are to give a relative sense of the weights on the final exam and have no bearing on extra credit points)

Optional, but due just after you take the final exam.

Problem 1: Discrete Mathematics (25 Points Total)

In Problem Set Three, you explored graphs and complement graphs. This problem explores binary relations and complementary binary relations.

If R is a binary relation over a set A, the complement relation R^c is the binary relation over A defined as follows:

$$\forall x \in A.\ \forall y \in A.\ (xR^cy \leftrightarrow \neg(xRy))$$

In other words, xR^cy is true iff xRy is false. For example, the complement of the = relation is the $\neq$ relation, and the complement of the $\leq$ relation is the $>$ relation.

(i) Complementary Relations, Part One (15 Points)

Prove or disprove: if R is a binary relation over a nonempty set A, then ***at least*** one of R or R^c is a partial order.

(ii) Complementary Relations, Part Two (10 Points)

Prove or disprove: if R is a binary relation over a nonempty set A, then ***at most*** one of R or R^c is a partial order.

Problem 2: Regular Languages (45 Points Total)

Consider the following language over $\Sigma = \{ \texttt{O}, \texttt{E} \}$:

$$PARITY = \{ w \mid w \text{ has even length and has the form } \texttt{E}^n \text{ or } w \text{ has odd length and has the form } \texttt{O}^n \}$$

For example, $\texttt{EE} \in PARITY$, $\texttt{OOOOO} \in PARITY$, $\texttt{EEEE} \in PARITY$, and $\varepsilon \in PARITY$, but $\texttt{EEE} \notin PARITY$, $\texttt{EO} \notin PARITY$, and $\texttt{OOOO} \notin PARITY$.

(i) Regular Expressions **(10 Points)**

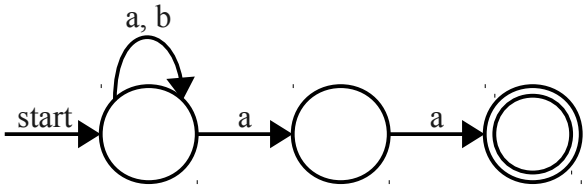
Write a regular expression for *PARITY*.

(ii) Finite Automata **(10 Points)**

Design a DFA that accepts *PARITY*.

(iii) NFAs **(10 Points)**

Use the subset construction to convert this NFA into an equivalent DFA:



(iv) Nonregular Languages **(15 Points)**

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

$$TWICE = \{ ww \mid w \in \Sigma^* \}$$

For example, $\texttt{0101} \in TWICE$, $\texttt{001001} \in TWICE$, $\texttt{1111} \in TWICE$, and $\varepsilon \in TWICE$, but $\texttt{01} \notin TWICE$.

Prove that *TWICE* is not regular.

Problem 3: Context-Free Languages (15 Points)

On Problem Set 5 and 6, you explored the language *ADD* over the alphabet { **1**, **+**, **=** }, which was defined as follows:

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

Consider the following generalization of *ADD*, which we will call *MULTIADD*, which consists of all strings describing unary encodings of two sums that equal one another. For example:

1 + 3 = 4	would be encoded as	**1+111=1111**
4 = 1 + 3	would be encoded as	**1111=1+111**
2 + 2 = 1 + 3	would be encoded as	**11+11=1+111**
2+0+2+0=0+4+0	would be encoded as	**11++11+=+1111+**
0=0	would be encoded as	**=**

Notice that there can be any number of summands on each side of the **=**, but there should be exactly one **=** in the string; thus **1=1=1** $\notin$ *MULTIADD*.

Write a CFG that generates *MULTIADD*. Show a parse tree for **1+1=11+** and **+=+**.

Problem 4: R and RE Languages (55 Points Total)

(i) Same Difference? (30 Points)

Prove or disprove: If $L_1 \in \mathbf{R}$ and $L_2 \in \mathbf{R}$, then $L_1 - L_2 \in \mathbf{R}$.

Prove or disprove: If $L_1 \in \mathbf{RE}$ and $L_2 \in \mathbf{RE}$, then $L_1 - L_2 \in \mathbf{RE}$.

(ii) `a*b` is Undecidable? (10 Points)

All regular languages are decidable, but below is a purported proof that the regular language described by the regular expression **`a*b`** is undecidable:

> *Theorem:* **`a*b`** is undecidable.
>
> *Proof:* By contradiction; assume **`a*b`** is decidable. Let D be a decider for it. Now, consider what happens when we run D on a string of infinitely many **`a`**'s followed by a **`b`** and a string of infinitely many **`a`**'s. Let's call this first string x and the second string y. Since D is a decider, it halts on all inputs, and therefore cannot run for an infinitely long time. Therefore, D must halt before reading the last character of x and the last character of y. Because x and y are the same except for their last character, we see that D must have the same behavior when run on x and when run on y. If D accepts x, then D also accepts y, but y is not in the language **`a*b`**. Otherwise, D rejects x, but x is in the language **`a*b`**. Both cases contradict the fact that D is a decider for **`a*b`**. We have reached a contradiction, so our assumption must have been wrong. Thus **`a*b`** is undecidable. ■

What's wrong with this proof?

(iii) Accept Most of the Strings! (15 Points)

Let $A_{MOST} = \{ \langle M, n \rangle \mid M \text{ is a TM, } n \in \mathbb{N}, \text{ and } M \text{ accepts all strings of length at least } n \}$. Prove that A_{MOST} is neither **RE** nor co-**RE**. As a hint, you may want to use a mapping reduction involving the language A_{ALL} from Problem Set 8. As a reminder:

$$A_{ALL} = \{ \langle M \rangle \mid M \text{ is a TM and } \mathscr{L}(M) = \Sigma^* \}$$

Problem 5: P and NP (40 points total)

(i) Closure under Complement (20 Points)

Prove that **P** is closed under complementation. *(Hint: Show how to turn a polynomial-time decider for a language L into a polynomial-time decider for the language $\overline{L}$)*

While we know that **P** is closed under complementation, it is unknown whether **NP** is closed under complementation. The class of problems that are the complements of problems in **NP** is an interesting one, and it is so important that we give it the name *co-**NP***. Formally, co-**NP** is the set of languages L such that $\overline{L} \in$ **NP**. For example, the language

$$\text{SAT} = \{ \langle \varphi \rangle \mid \varphi \text{ is a satisfiable propositional logic formula} \}$$

is known to be in **NP**, while its complement

$$\overline{\text{SAT}} = \{ \langle \varphi \rangle \mid \varphi \text{ is an unsatisfiable propositional logic formula} \}$$

is contained in co-**NP**.

Just as the relation between **P** and **NP** is unknown, the relation between **NP** and co-**NP** is also unknown and is a major open problem in complexity theory. However, we do know of one interesting result about how **P**, **NP**, and co-**NP** are connected.

(ii) NP and co-NP (10 Points)

Prove that if **NP** ≠ co-**NP**, then **P** ≠ **NP**.

(iii) What Do We Know? **(10 Points)**

Below are ten statements, some of which are definitely known to be true, some of which are definitely known to be false, and some of which aren't known to be true or false. For each of these statements, write a **T** if the statement is definitely true, an **F** if the statement definitely false, and a **?** if neither of these are the case.

For any language L, if $L \in \mathbf{P}$, then $L \in \mathbf{NP}$. _____

For any language L, if $L \in \mathbf{NP}$, then $L \in \mathbf{P}$. _____

For any languages L and L', if L is **NP**-complete and $L' \leq_P L$, then $L' \in \mathbf{P}$. _____

For any languages L and L', if L is **NP**-complete and $L' \leq_P L$, then $L' \in \mathbf{NP}$. _____

For any languages L and L', if L is **NP**-complete and $L' \leq_P L$, then $L' \in \mathbf{NPC}$. _____

If 3SAT is decidable in time $O(n^{10})$, then $\mathbf{P} = \mathbf{NP}$. _____

If 3SAT is not decidable in time $O(n^{10})$, then $\mathbf{P} \neq \mathbf{NP}$. _____

There exists an **NP** language that is not in **R**. _____

There exists an ***NP-complete*** language that is not in **R**. _____

There exists an ***RE-complete*** language that is in **NP**. _____