Extra Credit Practice Final

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 with questions. It is perfectly fine to work on these problems in a group or even to ask questions about them at the review session, but I strongly suggest taking this practice exam under exam conditions.

The 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.

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 this final exam. There will be 180 total points, which corresponds to roughly one point per question. The exam will be worth 25% of your total grade in this course.

Question		**Points**	**Grader**
(1) Induction	(20)	/20	
(2) Regular Languages	(25)	/25	
(3) Context-Free Languages	(25)	/25	
(4) **R** and **RE** Languages	(70)	/70	
(5) **P** and **NP**	(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 before you take the final exam.

Problem 1: Induction (20 points total)

Recall that we can compactly express large sums using the notation

$$\sum_{i=1}^{n} a_i = a_1 + a_2 + \ldots + a_n$$

We can use similar notation for products:

$$\prod_{i=1}^{n} a_i = a_1 \cdot a_2 \cdot \ldots \cdot a_n$$

Just as the empty sum of no numbers is defined to be 0, the empty product of no numbers is defined to be 1.

Consider a sequence of $n + 1$ real numbers $x_0, x_1, x_2, \ldots, x_n$, where $n \geq 0$ and each $x_i > 0$. Prove, by induction, that

$$\prod_{i=1}^{n} \frac{x_i}{x_{i-1}} = \frac{x_n}{x_0}$$

Problem 2: Regular Languages (25 points total)

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

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

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

(i) Regular Expressions (8 Points)

Write a regular expression for *PARITY*.

(ii) Finite Automata (7 Points)

Design a DFA that accepts *PARITY*.

(iii) The Pumping Lemma **(10 Points)**

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

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

For example, `0101` $\in$ *TWICE*, `001001` $\in$ *TWICE*, `1111` $\in$ *TWICE*, and $\varepsilon \in$ *TWICE*, but `01` $\notin$ *TWICE*.

Using the pumping lemma for regular languages, prove that *TWICE* is not regular.

Problem 3: Context-Free Languages (25 points total)

(i) Designing CFGs **(10 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*.

ii) Designing DPDAs **(15 Points)**

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` $\in$ *LE*, `11` $\in$ *LE*, $\varepsilon \in$ *LE*, but `00011` $\notin$ *LE* and `01110` $\notin$ *LE*.

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 (70 points total)

(i) Same Difference? (12 Points)

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

(ii) Not the Same Difference? (13 Points)

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

(iv) Accept Most of the Strings! (25 Points)

Consider the language

$$\mathrm{A}_{\mathrm{MOST}} = \{ \langle M, n \rangle \mid \text{M accepts all strings of length at least } n \}$$

Prove that $\mathrm{A}_{\mathrm{MOST}}$ is undecidable by reducing A_{TM} to it.

(v) Accept Most of the Strings! (Take Two) (20 Points)

Prove that $\mathrm{A}_{\mathrm{MOST}}$ is unrecognizable by reducing $\mathrm{A}_{\mathrm{ALL}}$ to it. As a reminder:

$$\mathrm{A}_{\mathrm{ALL}} = \{ \langle \mathrm{M} \rangle \mid \mathscr{L}(M) = \Sigma^* \}$$

Problem 5: P and NP (45 points total)

(i) Closure under Complement (15 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

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

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

$$\overline{SAT} = \{ \varphi \mid \varphi \text{ is not a syntactically correct propositional logic formula, or } \varphi \text{ is unsatisfiable } \}$$

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** $\neq$ co-**NP**, then **P** $\neq$ **NP**.

(iii) What Do We Know? (20 Points)

Below are ten statements, some of which are definitely true, some of which are definitely false, and some of which are not necessarily true or false (either because the truth of the statement depends on the choice of some particular language, or because the statement depends on an open problem such as whether **P** = **NP**). For each of these statements, write a **T** if the statement is always true, an **F** if the statement is always false, and a **?** if with what is provided the statement cannot be definitively shown to be true or false.

If $L \in \mathbf{P}$, then $L \in \mathbf{NP}$. _____

If $L \in \mathbf{NP}$, then $L \in \mathbf{P}$. _____

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

If L is **NP**-complete and $L' \leq_P L$, then $L' \in \mathbf{NP}$. _____

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 **P** = **NP**. _____

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

There exists an **NP** language that is not **RE**. _____

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

There exists an **NP**-***hard*** language that is not **RE**. _____