Practice CS103 Final Exam

This practice exam is closed-book and closed-computer but open-note. You may have a double-sided, 8.5" × 11" sheet of notes with you when you take this exam. Please hand-write all of your solutions on this physical copy of the exam.

> On the actual exam, there'd be space here for you to write your name and sign a statement saying you abide by the Honor Code. We're not collecting or grading this practice exam (though you're welcome to step outside and chat with us about it when you're done!) and this exam doesn't provide any extra credit, so we've opted to skip that boilerplate.

You have three hours to complete this practice exam. There are 48 total points. This practice exam is purely optional and will not directly impact your grade in CS103, but we hope that you find it to be a useful way to prepare for the final exam. You may find it useful to read through all the questions to get a sense of what this exam contains before you begin.

Question	**Points**	**Graders**
(1) Relations and Functions	/ 6	
(2) Graphs and Induction	/ 6	
(3) Regular Languages	/ 12	
(4) Context-Free Languages	/ 4	
(5) **R**, **RE**, and co-**RE** Languages	/ 17	
(6) **P** and **NP** Languages	/ 3	
	/ 48	

Good Luck!

Problem One: Relations and Functions (6 Points)

Below is a series of statements, each of which is true or false. For each statement, decide whether it's true or false. No justification is necessary.

i. **(1 Point)** There is at least one binary relation that is both a total ***relation*** and an equivalence relation.

ii. **(1 Point)** There is at least one binary relation that is both a total ***order*** and an equivalence relation.

iii. **(1 Point)** The binary relation $<$ over $\mathbb{R}$ is antisymmetric.

iv. **(1 Point)** If $f : \mathbb{R} \to \mathbb{R}$ is an injection, then $f(r) \geq r$ for all $r \in \mathbb{R}$.

v. **(1 Point)** If $f : \mathbb{N} \to \mathbb{N}$ is a bijection, then $f(n) = n$ for all $n \in \mathbb{N}$.

vi. **(1 Point)** If $f : \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$ are bijections, then the function $h : \mathbb{R} \to \mathbb{R}$ defined as $h(x) = f(x) + g(x)$ is also a bijection.

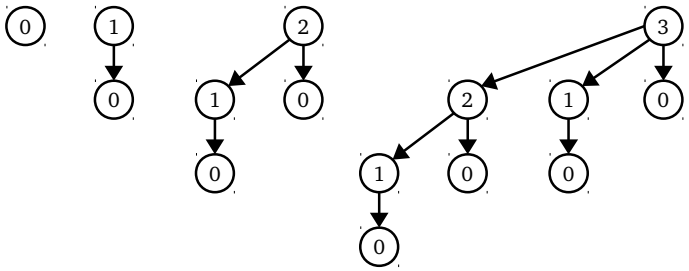
Problem Two: Graphs and Induction (6 Points)

Let's refresh some definitions from Problem Set Two. A ***directed tree*** is a special kind of directed graph with the following properties:

- There is a special node called the ***root node*** with no incoming edges.
- Every node other than the root node has exactly one incoming edge.

If a node u has an edge to a node v, we say that u is the ***parent*** of v. Similarly, v is a ***child*** of u.

Binomial trees are a specific family of directed trees defined as follows: a binomial tree of order n is a single node with n children, which are binomial trees of order 0, 1, 2, …, $n - 1$. For example, here are pictures of binomial trees of orders 0, 1, 2, and 3:



Here's an interesting observation: in a binomial tree of order n, the number of times the binomial tree of order 0 appears is

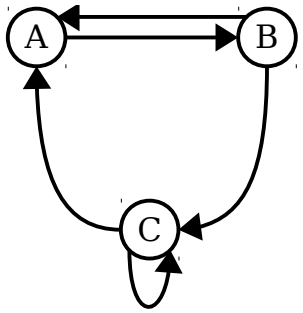
- 1, if $n = 0$, and
- 2^{n-1} otherwise

Prove, by induction, that this is the case.

Problem Three: Regular Languages (12 Points)

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

Consider the following graph *G*:



Let $\Sigma = \{\mathbf{A}, \mathbf{B}, \mathbf{C}\}$. 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**.

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

$\mathbf{A} \in L$	$\mathbf{ACC} \notin L$
$\mathbf{ABC} \in L$	$\varepsilon \notin L$
$\mathbf{BCC} \in L$	$\mathbf{BBA} \notin L$
$\mathbf{CCABA} \in L$	$\mathbf{ABBC} \notin L$

i. **(3 Points)** Design a DFA for *L*.

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

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

For example:

$\mathbf{1\leq 2} \in L$	$\mathbf{2\leq\leq} \notin L$
$\mathbf{1\leq 1\leq 2\leq 2} \in L$	$\mathbf{\leq 2} \notin L$
$\mathbf{2\leq 2\leq 2} \in L$	$\varepsilon \notin L$
$\mathbf{1\leq 1\leq 1\leq 1} \in L$	$\mathbf{1} \notin L$
$\mathbf{1\leq 1\leq 2} \in L$	$\mathbf{12\leq 22} \notin L$

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.

ii. **(2 Points)** Write a regular expression for L.

iii. **(1 Point)** Let L be an arbitrary language over Σ. Prove that if $x \in L$ and $y \notin L$, then x and y are distinguishable relative to L.

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

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

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.

iv. **(3 Points)** 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?)*

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

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

v. **(3 Points)** 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 Four: Context-Free Languages (4 Points)

On Problem Set 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*.

Problem Five: R, RE, and co-RE Languages (17 Points)

Let $L = \{ \langle M \rangle \mid M \text{ is a TM, } \mathscr{L}(M) \neq \emptyset, \text{ and } \mathscr{L}(M) \neq \Sigma^* \}$.

i. **(5 Points)** Prove that $L \notin$ **RE**.

Recall that a ***verifier*** for a language L is a TM V such that

- V halts on all inputs, and
- $\forall w \in \Sigma^*. (w \in L \quad \leftrightarrow \quad \exists c \in \Sigma^*.$ V accepts $\langle w, c \rangle)$

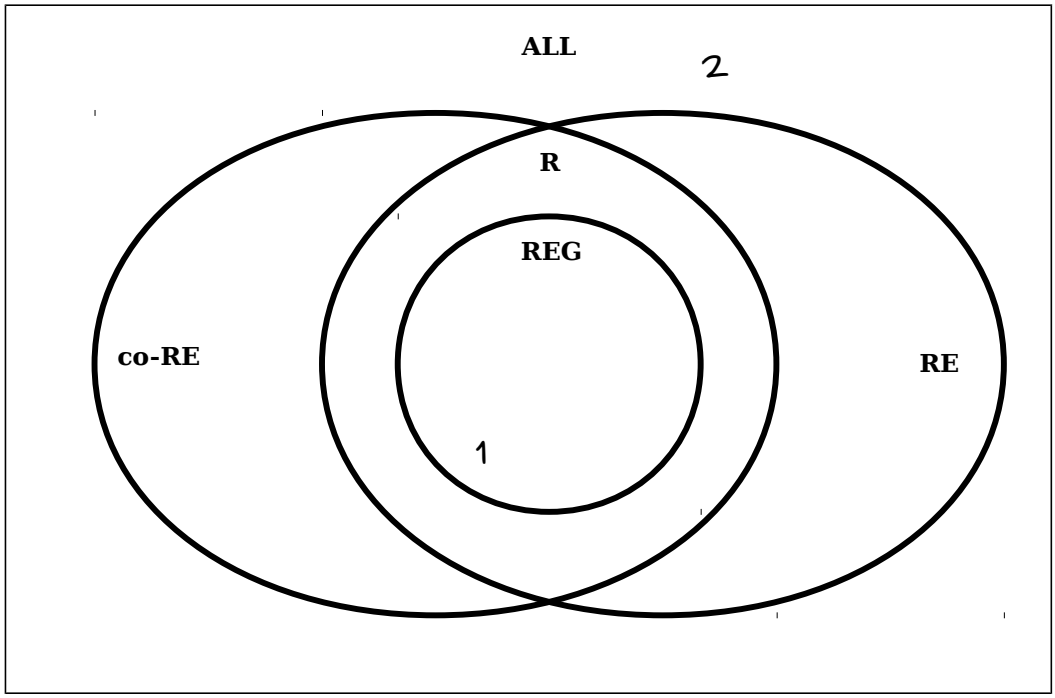
Let's say that a ***weak verifier*** for a language L is a TM X such that

- $\forall w \in \Sigma^*. (w \in L \quad \leftrightarrow \quad \exists c \in \Sigma^*.$ X accepts $\langle w, c \rangle)$

In other words, a weak verifier for a language L is like a normal verifier for L, except that it's not required to halt on all inputs.

ii. **(5 Points)** Prove that if X is a weak verifier for a language L, then $L \in$ **RE**.

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 nine 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 each language is worth exactly one point.



1. Σ^*
2. EQ_{TM}
3. $\{ a^n b^n c^n \mid n \in \mathbb{N} \}$
4. The concatenation of language (1) and language (3). Assume $\Sigma = \{ a, b, c \}$
5. The union of language (1) and language (3). Assume $\Sigma = \{ a, b, c \}$
6. $\{ \langle w_1, w_2 \rangle \mid$ either w_1 is the encoding of a TM M that accepts w_2 or w_2 is the encoding of a TM M that accepts $w_1 \}$
7. The complement of language (3).
8. The complement of language (6).
9. $\{ \langle M \rangle \mid M$ is a TM and $\mathscr{L}(M) = \emptyset \}$

Problem Six: P and NP Languages (3 Points)

Suppose that the following claim is true:

If L_1 and L_2 are **NP** languages other than Ø or Σ^*, then $L_1 \leq_p L_2$

Decide which of the following statements is true and briefly justify your answer:

- In this case, **P** is definitely equal to **NP**.
- In this case, **P** is definitely not equal to **NP**.
- In this case, **P** may or may not be equal to **NP**.