# Practice CS103 Final Exam V

***We strongly recommend that you work through this exam under realistic conditions rather than just flipping through the problems and seeing what they look like. Setting aside three hours in a quiet space with your notes and making a good honest effort to solve all the problems is one of the single best things you can do to prepare for this exam. It will give you practice working under time pressure and give you an honest sense of where you stand and what you need to get some more practice with.***

This practice final exam is essentially the final exam from Spring 2015, with one or two questions swapped out for questions from previous quarter's final exams. The sorts of questions here are representative of what you might expect to get on the upcoming final exam, though the point balance and distribution of problems might be a bit different.

The exam policies are the same for the midterms – closed-book, closed-computer, limited note (one double-sided sheet of 8.5" × 11" paper decorated however you'd like).

You have three hours to complete this exam. There are 49 total points.

| Question | Points | Graders |
|---|---|---|
| (1) Induction | / 4 | |
| (2) First-Order Logic | / 5 | |
| (3) Sets and Functions | / 5 | |
| (4) Regular and Context-Free Languages | / 16 | |
| (5) **R** and **RE** Languages | / 15 | |
| (6) **P** and **NP** Languages | / 4 | |
| | **/ 49** | |

## Problem One: Induction (4 Points)

If you're hungry, you can make yourself a cheese sandwich by taking two pieces of bread and putting a slice of cheese between them. If you're really hungry, you can make a *cheese metasandwich* by making two cheese sandwiches and putting a slice of cheese between them. If you're really, really hungry, you can make a *cheese meta-metasandwich* by making two cheese metasandwiches and putting a slice of cheese between them.

Formally, we can define a hierarchy of sandwich variants as follows:

- An ***order 0 metasandwich*** is a normal cheese sandwich.
- An ***order n+1 metasandwich*** consists of two order *n* metasandwiches with a piece of cheese between them.

Determine formulas for the number of pieces of bread and slices of cheese in an order *n* metasandwich, then prove by induction that your formulas are correct. Your formulas should not be recurrence relations, by the way – it should be easy to directly evaluate your formulas to see how much bread and cheese is necessary to make an order *n* metasandwich.

## Problem Two: First-Order Logic (5 Points)

*(Final Exam, Spring 2015)*

Suppose we have the predicates

- *Person*(*p*), which states that *p* is a person, and
- *Loves*(*p*, *q*), which states that *p* loves *q*.

Below are a series of five English statements paired with a statement in first-order logic. For each statement, decide whether the corresponding formula in first-order logic is a correct translation of the English statement and check the appropriate box. There is no penalty for an incorrect guess.

| | | |
|---|---|---|
| Everyone loves themselves. | $\forall p.\ (Person(p) \rightarrow \forall q.\ (Loves(p, q) \rightarrow p = q))$ | ☐ ***Correct*** ☐ ***Incorrect*** |
| There are two people that everyone loves. | $\forall r.\ (Person(r) \rightarrow \exists p.\ (Person(p) \wedge \exists q.\ (Person(q) \wedge q \neq p \wedge Loves(r, p) \wedge Loves(r, q))))$ | ☐ ***Correct*** ☐ ***Incorrect*** |
| Love is a transitive relation over the set of people. | $\forall p.\ (Person(p) \wedge \forall q.\ (Person(q) \wedge \forall r.\ (Person(r) \wedge (Loves(p, q) \wedge Loves(q, r) \rightarrow Loves(p, r)))))$ | ☐ ***Correct*** ☐ ***Incorrect*** |
| No two people love exactly the same set of people. | $\forall p.\ (Person(p) \rightarrow \forall q.\ (Person(q) \wedge q \neq p \rightarrow \exists r.\ (Person(r) \wedge (Loves(p, r) \leftrightarrow \neg Loves(q, r)))))$ | ☐ ***Correct*** ☐ ***Incorrect*** |
| Someone doesn't love anyone. | $\neg\forall p.\ (Person(p) \rightarrow \exists q.\ (Person(q) \wedge Loves(p, q)))$ | ☐ ***Correct*** ☐ ***Incorrect*** |

## Problem Three: Sets and Functions (5 Points)

*(Final Exam, Winter 2013)*

There can be many functions from one set $A$ to a second set $B$. This question explores how many functions of this sort there are.

For any set $S$, we will denote by $2^S$ the following set:

$$2^S = \{ f \mid f : S \to \{0, 1\} \}$$

That is, $2^S$ is the set of all functions whose domain is $S$ and whose codomain is the set $\{0, 1\}$. Note that $2^S$ does *not* mean "two raised to the $S$th power." It's just the notation we use to denote the set of all functions from $S$ to $\{0, 1\}$.

Prove that if $S$ is a nonempty set, then $|2^S| = |\wp(S)|$. To do so, find a bijection from $2^S$ to $\wp(S)$, then prove that your function is a bijection. Your proof should work for all sets $S$, including infinite sets.

You may find the following definition useful: if $f : A \to B$ and $g : A \to B$ are functions with the same domain and the same codomain, then we say that $f = g$ if $f(a) = g(a)$ for all $a \in A$.

*(Extra space for your answer to Problem Three, if you need it.)*

## Problem Four: Regular and Context-Free Languages (16 Points)

Let $\Sigma = \{\texttt{a}, \texttt{b}\}$ and let $L_1 = \{\ w \in \Sigma^* \mid w$ does not contain `bbb` as a substring $\}$.

i. **(3 Points)** Design a DFA for $L_1$.

ii. **(3 Points)** Write a regular expression for $L_1$.

*(Final Exam, Fall 2014)*

Let $\Sigma = \{a, b\}$ and consider the following language over $\Sigma$:

$$L_3 = \{\ w \in \Sigma^* \mid w \text{ has odd length and its middle character is } a\ \}$$

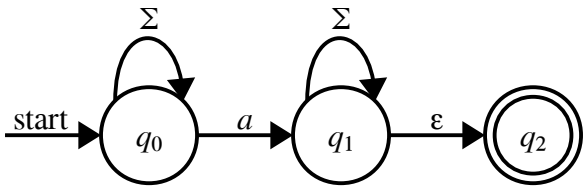
iii. **(5 Points)** Prove that $L_3$ is not a regular language.

*(Final Exam, Fall 2014)*

As a reminder, the language $L_3$ over $\Sigma = \{a, b\}$ from the previous page was defined as follows:

$$L_3 = \{ \ w \in \Sigma^* \mid w \text{ has odd length and its middle character is } a \ \}$$

You just proved that this language is not regular. However, below is an NFA that purportedly has language $L_3$:




Here is a line of reasoning that claims that this NFA has language $L$:

> "Intuitively, this NFA will sit in state $q_0$ following its Σ transition until it nondeterministically guesses that it's about to read the middle $a$ character. When it does, it transitions to $q_1$, where it keeps following the Σ transition as long as more characters are available. Finally, once it's read all the characters of the input, the NFA follows the ε transition from $q_1$ to $q_2$, where the NFA then accepts."

Of course, this reasoning has to be incorrect, since $L_3$ is not a regular language.

iv. **(2 Points)** Without using the fact that $L_3$ is not a regular language, explain why the above NFA is not an NFA for the language $L_3$.

*(Final Exam, Spring 2015)*

Let $\Sigma = \{\texttt{a}, \texttt{b}\}$ and consider the following language $L_5$ over $\Sigma$:

$$L = \{\ w \in \Sigma^* \mid |w| \equiv_3 0 \text{ and all the characters in the first third of } w \text{ are the same } \}$$

Here, aababa $\in L_5$, bbbaaaaaa $\in L_5$, aaa $\in L_5$, and $\varepsilon \in L_5$, but abbbbb $\notin L_5$ and aaaaa $\notin L_5$. (For convenience, I've underlined the first third of the characters in each string.)

v. **(3 Points)** Write a context-free grammar for $L$.

## Problem Five: R and RE Languages (15 Points)

*(Final Exam, Spring 2015)*

Let $\Sigma = \{ (, ) \}$. Consider the language $L = \{ w \in \Sigma^* \mid w$ is a string of balanced parentheses $\}$. For example, $(())) \in L$, $((()))(()()) \in L$, and $\varepsilon \in L$, but $))(( \notin L$, $()) \notin L$, and $(() \notin L$.

i. **(5 Points)** Design a TM that is a decider for the language $L$. Please draw out an actual TM consisting of states and transitions rather than providing a high-level description of the TM. No justification is necessary.

   Some hints:

   - Our solution doesn't use very many states. If you find yourself drawing out a huge TM, you might want to reevaluate your solution.
   - Rather than searching for an open parenthesis and trying to find the close parenthesis that matches it, instead search for a close parenthesis and work backwards to find the nearest open parenthesis.
   - Remember that if a state in a TM has no transition defined for a particular character in the tape alphabet, the TM will automatically reject if it reads that character in that state.

*(Final Exam, Spring 2015)*

Although we typically haven't treated the class **RE** as a set, it is indeed a set of languages, so we can speak of subsets of the **RE** languages.

Let $S \subseteq$ **RE** be an arbitrary subset of the **RE** languages where $\emptyset \in S$ and $\Sigma^* \notin S$. We can then define a new language $L_S$ as follows:

$$L_S = \{ \langle M \rangle \mid M \text{ is a TM and } \mathscr{L}(M) \in S \}$$

In other words, $L_S$ is the set of all TMs whose language is one of the languages included in set $S$.

ii. **(5 Points)** Prove that $L_S$ is not decidable. As a hint, think about the following self-referential program:

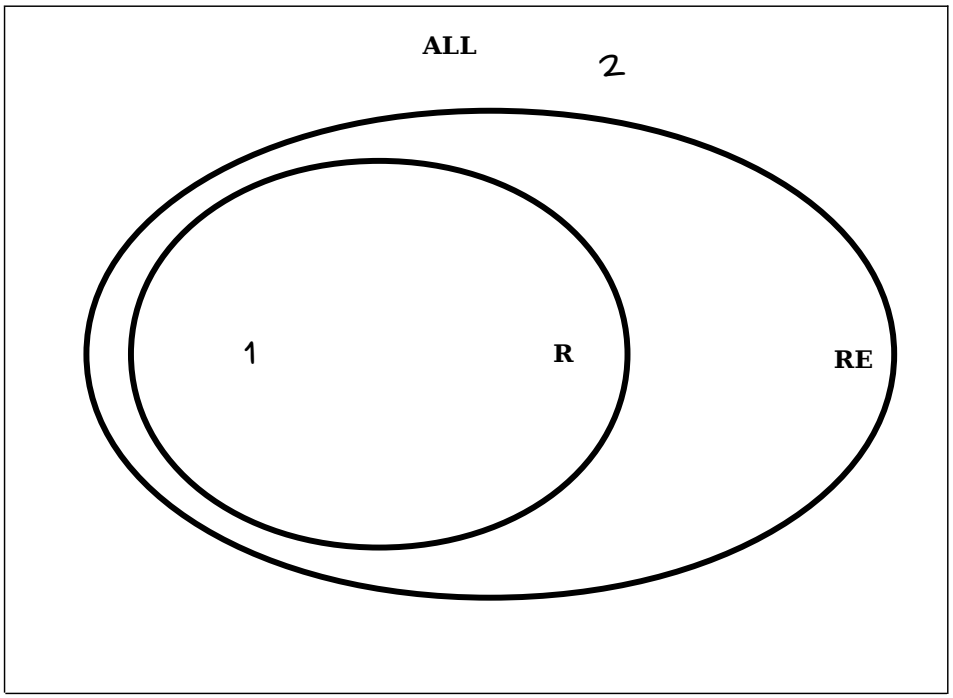
```
int main() {
    string me = mySource();
    string input = getInput();

    if (isInLS(me)) {
        accept();
    } else {
        reject();
    }
}
```

*(More space for Problem 5.ii, if you need it)*

*(Final Exam, Spring 2015)*

iii. **(5 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 seven 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 there is no penalty for an incorrect guess.




1. $\Sigma^*$
2. $L_D$
3. $\{ \langle D, w \rangle \mid D \text{ is a DFA and } D \text{ does not accept } w \}$
4. $\{ \langle R, w \rangle \mid R \text{ is a regular expression and } R \text{ does not match } w \}$
5. $\{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ does not accept } w \}$
6. $\{ \langle M \rangle \mid M \text{ is a TM and } M \text{ accepts } \langle M \rangle \}$
7. $\{ \langle M \rangle \mid M \text{ is a TM and there is no verifier for } \mathscr{L}(M) \}$

## Problem Six: P and NP Languages (4 Points)

*(Final exam, Fall 2011)*

Suppose that there is a language $L \in \mathbf{NP}$ where $L \notin \mathbf{P}$. Prove that in this case, no **NP**-complete language can be decided in polynomial time.