Foundations of Computer Science, CPSC 3500

Instructor: Prof. Ilya Safro, 228 McAdams Hall, Office hours: Thu, 12:15pm-1:15pm
TA: Ms. Joey Liu, email: xiaoyu3@g.clemson.edu

Course Structure

What	How many	Time	Points
Homework	Almost every class	1 week	10
Midterm exam I	1	1.25 hours	25
Midterm exam II	1	1.25 hours	25
Final exam	1	2.5 hours	30
Pop-up quizzes	Almost every class	5-7 min	10
Total			100.00

Points	Grade
≥ 89.00	A
≥ 79.00	B
≥ 65.00	C
≥ 55.00	D
≥ 0	F

Bonuses

Work in class, extra work in homework exercises, etc. - up to 10 points. We do not want to miss the next Turing, Fields and Nobel laureates, so any submitted conference/journal paper written during and as a result of this course - 100 points, and new interesting ideas - up to 100 points (both are based on instructor's subjective judgment).

NO ELECTRONIC DEVICES IN THIS CLASSROOM!

All slides will be available online after each class

(Midterm I + Midterm II)/2 ≥ 90 (before curving, if any)

+

Average over all homework assignments ≥ 90

+

Each quiz ≥ 70

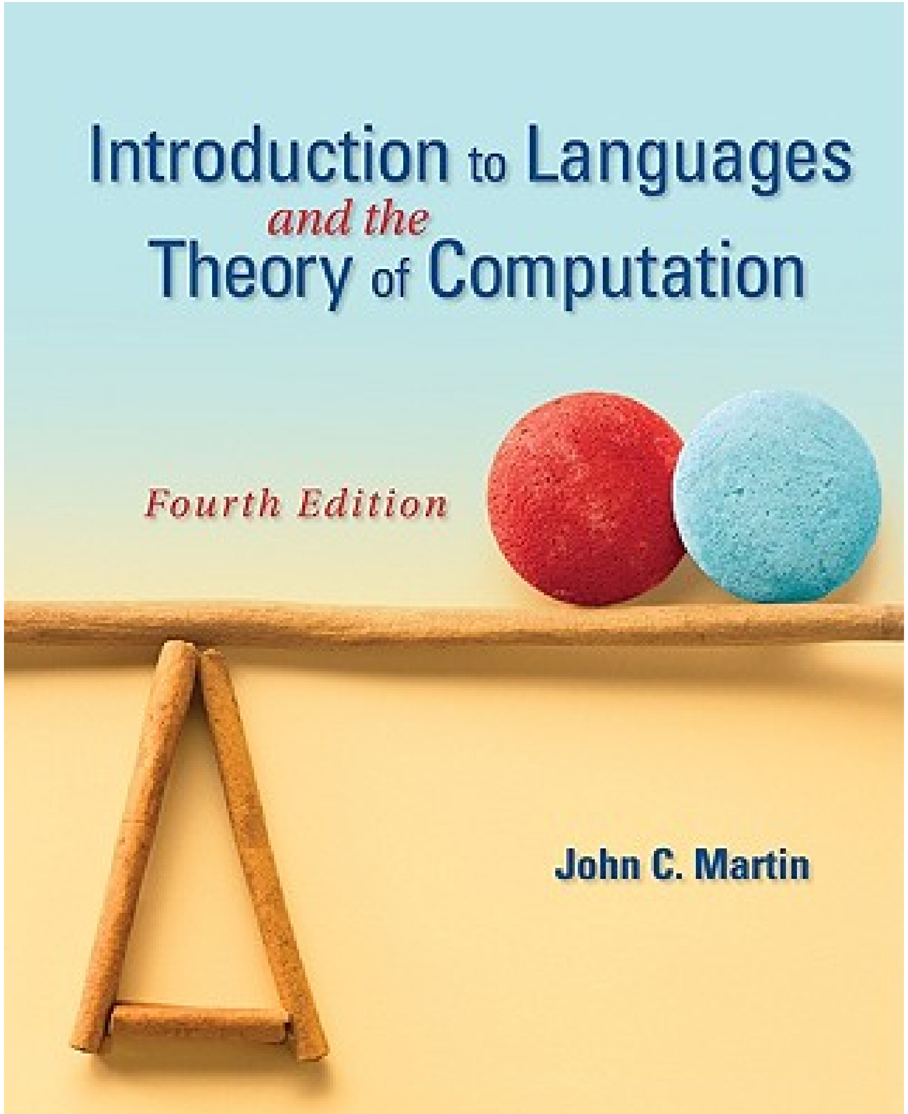
(which also means the attendance)

=

No final exam

- **Pop-up quizzes: no unexcused absences are allowed; unexcused absences get counted as zeros**
- **<u>If</u> any curving will be used, a minimum score of 50 is still required to pass this course**

Recommended Book



Assumption:
You have all prerequisites and you know

- mathematical induction
- basics of set theory (sets, inclusion, difference, union, **proofs of equality**, etc.)
- basics of mathematical logic (operators and/or/not/→, **proofs “if and only if”**, etc.)

You can find this material in Chapter I.

- Grimaldi “Discrete and Combinatorial Mathematics: An Applied Introduction “ (very good introductory book)
- Linz “Formal Languages and Automata” (not easy)
- Goddard “Theory of Computation” (perfect for concepts)
- Hopcroft, Motwani, Ullman “Introduction to Automata Theory, Languages, and Computation” (not easy, but it is considered as one of the best books in FOCS)
- Meduna “Automata and Languages”
- Sudkamp “Languages and Machines”

- http://www.jflap.org/

JFLAP is a software for experiments with formal languages
Use it when we will start with finite automata (in 1-2 weeks)

Important

- In some homework assignments, new definitions, principles, and algorithms will be introduced. Exams can include them! All exams are cumulative over any and all previous and current material.
- Exams will be closed book, closed notes and closed any other aids. A score of 0 will be given to anyone not present at the beginning of the exam.

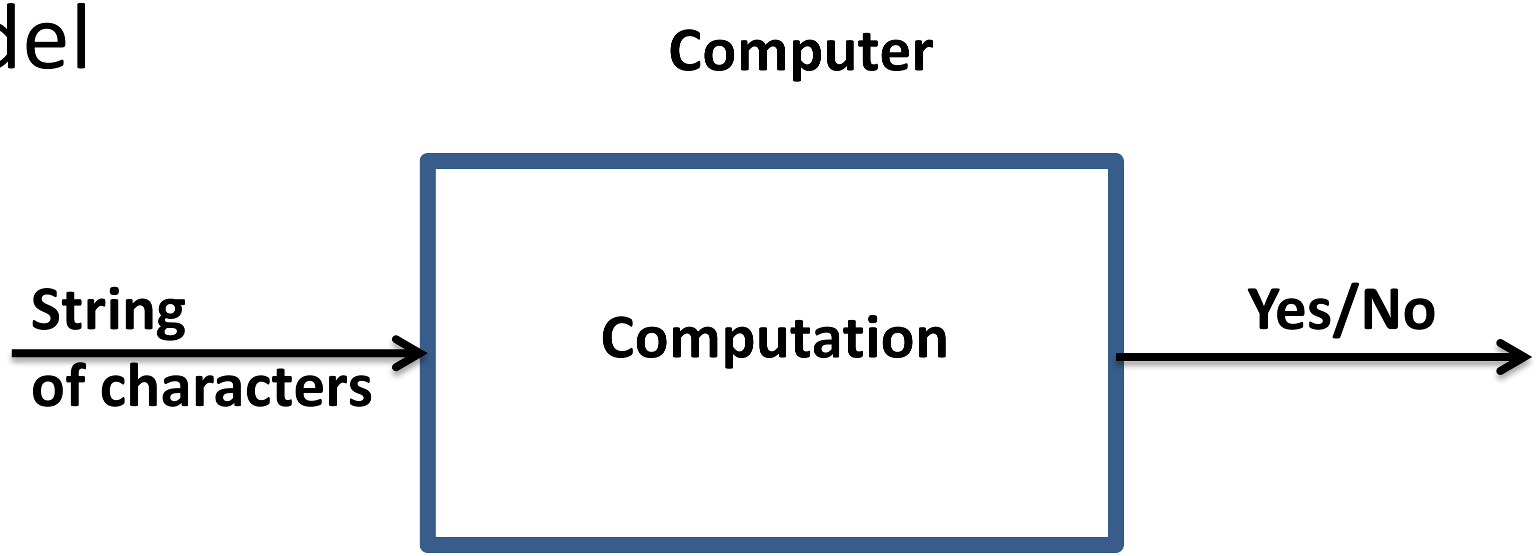
Very important

- You cannot copy-paste solutions from the Internet, books, friends, etc. If you don’t solve them by yourself, this will be the best way to fail the exams.
- Solve as many exercises from the textbook as you can. Try to solve more than what you get in the assignments. **This is not a passive learning course.**
- **Common mistake: you are sure that you understand some chapter (which is easy) but you did not solve 30-40 problems from that chapter by yourself (which is hard).**
- All chapters are cumulative. Do not neglect any material.

Chapter 1

Languages

Model



This computer plays a role of a language acceptor

Examples:

- We submit a string and ask whether this string is a correct algebraic expression or not.

  a+b*a → the answer is YES

  aa+++b--- → the answer is NO

- We submit a string and ask whether this string includes exactly 3 characters *a,* and 5 characters *b* or not.

Definition (Alphabet)

- An alphabet (usually denoted by Σ) is a finite set of symbols, such as $\{a, b\}$ or $\{0, 1\}$ or $\{A, B, C, \ldots, Z\}$ or $\{\spadesuit, \bigstar, \heartsuit\}$.
- A string over Σ is a finite sequence of symbols in Σ. For a string x, $|x|$ stands for the length (the number of symbols) of x.
- $n_\sigma(x)$ is the number of occurrences of the symbol σ in the string x. Example: $n_a(abbba) = 2$.
- The null string Λ is a string over Σ, no matter what the alphabet Σ is. By definition, $|\Lambda| = 0$.
- The set of all strings over Σ will be written Σ^*. For $\Sigma = \{a, b\}$, we have $\{a, b\}^* = \{\Lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$

canonical order

Remark: There exist infinite alphabets but not in this course.

Definition A language over Σ is a subset of Σ^*.

Examples:

empty set

- The empty language $\emptyset$.
- $\{\Lambda, a, aab\}$
- The language PAL of palindromes over $\{a, b\}$ (strings such as aba or baab that are unchanged when the order of the symbols is reversed).
- $\{x \in \{a, b\}^* \mid n_a(x) > n_b(x)\}$
- $\{x \in \{a, b\}^* \mid |x| \geq 2 \text{ and } x \text{ begins and ends with } b\}$

Definition A language over Σ is a subset of Σ^*.

Examples:

empty set

- The empty language $\emptyset$.
- $\{\Lambda, a, aab\}$
- The language PAL of palindromes over $\{a, b\}$ (strings such as aba or baab that are unchanged when the order of the symbols is reversed).
- $\{x \in \{a, b\}^* \mid n_a(x) > n_b(x)\}$
- $\{x \in \{a, b\}^* \mid |x| \geq 2 \text{ and } x \text{ begins and ends with } b\}$

Λ is always an element of Σ^*, but other languages over Σ may or may not contain it. More real examples:

- The language of numeric literals in Java such as 0.3 and 5.0E-3.
- The language of legal Java programs. $\Sigma = \{\text{numbers,letters}, \ldots\}$.

- Concatenation: if x and y are two strings, the concatenation of x and y is written xy and consists of the symbols of x followed by those of y. Example: if $x = ab$ and $y = bab$ then $xy = abbab$ and $yx = babab$.
  For every string x, $x\Lambda = \Lambda x = x$.

- If s is a string and $s = tuv$ for three strings t , u, and v, then t is a prefix of s, v is a suffix of s, and u is a substring of s.

- Concatenation of languages L_1 and L_2 is the language

$$L_1 L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$$

  Example: $\{a, aa\}\{\Lambda, b, ab\} = \{a, ab, aab, aa, aaab\}$.

- Exponents: if $a \in \Sigma$ $a^k = aa \ldots a$ (k times); if $x \in L$

$$x^k = xx \ldots x$$

  similar for L^k, where L is a language.

- For pairs of alphabets and languages we can define union, intersection, and difference operations ($\cup, \cap, -$)

$$\begin{aligned}
\Sigma_1 \cup \Sigma_2 &= \{a \mid a \in \Sigma_1 \text{ or } a \in \Sigma_2\} \\
\Sigma_1 \cap \Sigma_2 &= \{a \mid a \in \Sigma_1 \text{ and } a \in \Sigma_2\} \\
\Sigma_1 - \Sigma_2 &= \{a \mid a \in \Sigma_1 \text{ and } a \notin \Sigma_2\}
\end{aligned}$$

  Same for languages L_1, and L_2.

- The Kleene star (or Kleene closure) operation on a language L

$$L^* = \bigcup \{L^k \mid k \in \mathbb{N}\}, \text{ where } L^0 = \{\Lambda\}$$

- Precedence rules are similar to the algebraic rules. Example:

$$L_1 \cup L_2 L_3^* = L_1 \cup (L_2(L_3^*))$$

Mathematical Induction and Recursion

Prove by induction on n that

$$\sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2}$$

Mathematical Induction: Reminder

- Basis step: $n = 1$, $1 = 1(1+1)/2$
- Hypothesis: Suppose the claim is true for some $n = k$,

$$\sum_{i=1}^{k} i = k(k+1)/2$$

- Induction step: We need to prove the claim for $n = k + 1$

$$\sum_{i=1}^{k+1} i = \sum_{i=1}^{k} i + (k+1) = k(k+1)/2 + (k+1) = (k+1)(k+2)/2$$

Thus the claim is true for all n.