The second exam will be given in class on Monday, March 15. If you have an unavoidable conflict with the date of an exam, please see me as soon as possible (before the exam date!) to discuss options for rescheduling. Last minute rescheduling/extensions will not be accommodated for something known about in advance.

The exam covers chapter 3 (except section 3.3) and the first part of chapter 4 (sections 4.1-4.2). Sections 4.3-4.4 (parsing, parse trees, and pushdown automata) will not be on this exam. You will also not be asked about `grep` *or other Linux commands, the syntax used in applications of regular expressions, or regular expressions in Java.*

Many of the questions on the exam will be similar to problems on the homeworks. There many be some "short essay" questions that ask you to define something, or discuss something, or explain something, and so on. There might be some simple proofs about regular languages and context-free languages, or use of the Pumping Lemma.

Terms and ideas that you should be familiar with:

- alphabet (finite, non-empty set of "symbols")
- string over an alphabet Σ
- length of a string, $|x|$
- empty string, ϵ
- concatenation of strings, xy or $x \cdot y$
- reverse of a string, x^R
- x^n, for a string x and a natural number n
- $n_\sigma(x)$, the number of occurrences of a symbol σ in a string x
- the set of all possible strings over Σ, denoted Σ^*
- language over an alphabet Σ (a subset of Σ^*)
- a language over Σ is an element of $\mathcal{P}(\Sigma^*)$
- the set of strings over Σ is countable; the set of languages over Σ is uncountable
- union, intersection, set difference, and complement applied to languages
- concatenation of languages: LM, L^n for $n \in N$
- Kleene-star operation on a language: L^*
- regular expression over an alphabet Σ; the operators: $*$, $|$, and concatenation
- regular language; the language $L(r)$ generated by a regular expression r
- DFA (Deterministic Finite Automaton)
- transition diagram [the usual picture] of a DFA
- state (in a finite-state automaton); start state; accepting state (also known as final state)
- definition of a DFA as a list of five things, $(Q, \Sigma, q_o, \delta, F)$ — and what each thing means
- how a DFA computes (that is, what it does when it reads and processes a string)

- NFA (Non-deterministic Finite Automaton); the differences between NFAs and DFAs
- nondeterminism
- ϵ-transitions
- what it means for an NFA to accept a string
- the language, $L(M)$, accepted by an NFA or DFA M
- algorithm for converting an NFA to an equivalent DFA
- algorithm for converting a regular expression to an equivalent NFA
- DFAs, NFAs, and regular expressions all define the same class of languages
- operations ($L \cup M$, $L \cap M$, LM, L^*, $\overline{L}$, L^R) on regular languages produce regular languages
- the Pumping Lemma and showing that a language isn't regular
- CFGs (Context-Free Grammars)
- production rules; non-terminal and terminal symbols; start symbol
- definition of a CFG as a list of 4 things, $G = (V, \Sigma, P, S)$
- derivation (of a string from the start symbol of a CFG)
- the language, $L(G)$, generated by a CFG G; context-free language
- if L and M are context-free languages, then so are $L \cup M$, LM, and L^*
- every regular language is context-free
- BNF (Backus-Naur Form)
- using BNF to define the syntax of a language

- examples of languages that are not regular, such as:
  - $\{\ a^n b^n \mid n = m\ \}$
  - $\{\ a^n b^m c^k \mid k = n + m\ \}$
  - $\{\ w \in \{a, b\}^* \mid w = w^R\ \}$
  - $\{\ w \in \{a, b\}^* \mid n_a(w) < n_b(w)\ \}$
  - $\{\ ww \mid w \in \{a, b\}^*\ \}$
  - $\{\ a^n b^n c^n \mid n \in N\ \}$
  - $\{\ a^{n^2} \mid n \in N\ \}$
  - $\{\ www \mid w \in \{a, b\}^*\ \}$
- examples of languages that are context-free but not regular, such as the first four languages in the previous list

- some of the tasks that you could be asked to perform:
  - finding the language generated by a given regular expression
  - finding a regular expression for a given language
  - finding a DFA or NFA for a given language
  - finding a regular expression for an NFA or DFA
  - converting an NFA into an equivalent DFA, using the algorithm
  - converting a regular expression into an equivalent NFA, using the algorithm
  - determining whether a given string is accepted by an NFA or DFA

- finding a derivation for a given string from a given CFG
- finding a CFG for a given language
- finding the language generated by a given CFG
- using BNF in basic ways