The final exam takes place during the officially scheduled period, Tuesday, December 15, from 7:00 PM to 10:00 PM. The exam is in our regular classroom, Eaton 111.

The test will be about six pages long, and should take most people in the class less than two hours to complete. The last page will consist of one or two summary essay questions that ask you to bring together various things that have been covered in the course and to show your understanding of the basic ideas and overall themes. Questions on the other five pages will be similar to the questions on previous exams. Roughly half of those five pages will be devoted to material that we have covered since the third test, with the rest on material from the first three tests.

*The exam will **not** include a pumping lemma proof or a proof by induction. There will, however, be some proofs, possibly including a formal proof of the validity of an argument. The will be nothing related to logic circuits or to real computers (Java, BNF, regular expressions on computers). You will not be asked to create a pushdown automaton. You might be asked to create a general grammar for some language, and you might be asked to draw a transition diagram for a simple Turing machine. You will certainly need to know how to execute a Turing machine or pushdown automaton.*

Here are some terms and ideas from the last segment of the course:

Parsing
Parse tree for a string in a context-free language
Left derivation for a string in a context-free language
Pushdown automaton
How a pushdown automaton uses the stack
The language defined by a pushdown automaton
Transition diagram for a pushdown automaton
General grammars
Derivations using a general grammar
How a general grammar generates a language
Turing machine
Halt state
Transition diagram for a Turing machine
Table of rules for a Turing machine
How a Turing machine operates
Evidence that Turing machines are general-purpose computing devices
Turing acceptable language, also known as a recursively enumerable language
Turing decidable language, also known as a recursive language
A language L is recursive if and only if both L and $\overline{L}$ are recursively enumerable
The language hierarchy: finite, regular, context-free, recursive, recursively enumerable
The Church-Turing Thesis
How to make a list of all Turing machines (up to equivalence)
Turing machines (up to equivalence) can be assigned numbers: T_o, T_1, T_2, T_3, ...
The set K = $\{n \mid T_n$ halts when run with input $n\}$
The set K is recursively enumerable, but it is not recursive
The set $\overline{K}$ is not recursively enumerable
Universal Turing Machine
The Halting Problem and its computational unsolvability
The nature of computation

Here are some things from earlier in the course:

translations from logic to English, and from English to logic
propositional logic
proposition
the logical constants $\mathbb{T}$ and $\mathbb{F}$
the logical operators "and" ($\wedge$), "or" ($\vee$), and "not" ($\neg$)
truth table
logical equivalence ($\equiv$)
the conditional or "implies" operator ($\rightarrow$)
equivalence of $p \rightarrow q$ and $(\neg p) \vee q$
the negation of $p \rightarrow q$ is equivalent to $p \wedge \neg q$
the biconditional operator ($\leftrightarrow$)
equivalence of $p \leftrightarrow q$ with $p \rightarrow q \wedge q \rightarrow p$
tautology
some basic laws of Boolean algebra (double negation, De Morgan's, commutative, etc.)
converse of an implication
contrapositive of an implication
logical equivalence of an implication and its contrapositive
predicate logic
quantifiers, "for all" ($\forall$) and "there exists" ($\exists$)
negation of a statement that uses quantifiers
the difference between $\forall x\, \exists y$ and $\exists y\, \forall x$
arguments, valid arguments, and deduction
formal proof of the validity of an argument
Modus Ponens and Modus Tollens
translating arguments between English and logic
existence proof
counterexample
proof by contradiction
sets
set notations: $\{a, b, c\}$, $\{1, 2, 3, \dots\}$, $\{x \mid P(x)\}$, $\{x \in A \mid P(x)\}$; the empty set, $\emptyset$ or $\{\,\}$
element of a set: $a \in A$; subset: $A \subseteq B$
$A = B$ if and only if both $A \subseteq B$ and $B \subseteq A$
union, intersection, and set difference: $A \cup B$, $A \cap B$, $A \setminus B$
definition of set operations in terms of logical operators
power set of a set: $\mathcal{P}(A)$
universal set; complement of a set (in a universal set): $\overline{A}$
ordered pair: (a, b); cross product of sets: $A \times B$
one-to-one correspondence
cardinality of a finite set: $|A|$
finite set (in one-to one correspondence with one of the sets $N_0, N_1, N_2, \dots$)
infinite set (not finite)
countably infinite set (in one-to-one correspondence with $\mathbb{N}$)
a set is countably infinite iff its elements can be placed into an infinite list
uncountable set (that is, uncountably infinite)
the union of two countably infinite sets is countably infinite
for any set A, there is no one-to-one correspondence between A and $\mathcal{P}(A)$
alphabet (finite, non-empty set of "symbols")
string over an alphabet Σ
string operations: length ($|w|$), concatenation (xy), reverse (w^R), w^n, $n_\sigma(w)$

language over an alphabet Σ (a subset of Σ^*)
the set of strings over Σ is countable; the set of languages over Σ is uncountable
set operations on languages: union, intersection, set difference, and complement
concatenation (LM), power (L^n), and Kleene start (L^*) of languages
regular experssion over an alphabet Σ
regular language; the language $L(r)$ generated by a regular expression r
DFA (Deterministic Finite Automaton)
transition diagram of a DFA
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)
what it means for a DFA to accept a language
nondeterminism; NFA (Non-deterministic Finite Automaton)
what it means for an NFA to accept a string
the language, $L(M)$, accepted by a DFA or NFA
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
CFGs (Context-Free Grammars)
definition of a CFG as a 4-tuple $G = (V, \Sigma, P, S)$
derivation (of a string from the start symbol of a CFG)
context-free language
if L and M are context-free languages, then so are $L \cup M$, LM, and L^*

Here are some sample questions, taken from old final exams:

*Note: This is **absolutely not** meant as a comprehensive review!*

1. Use a *truth table* to show that $(p \to q) \wedge ((\neg p) \to q)$ is logically equivalent to q. (State what it is about your table that proves logical equivalence.)

2. Simplify each of the following logical expressions, so that the logical `NOT` operation applies only to simple terms:

a) $\neg(p \vee q \vee (\neg r))$

b) $\neg\big(\forall x\, \exists y\, \big(L(x, y) \wedge \forall z(\neg L(x, z))\big)\big)$

c) $\neg\big(\forall x\, (P(x) \to (\exists y\, (R(y) \wedge Q(x, y))))\big)$

3. English and logic...

a) State the negation of the statement, "If I do well in CS 229, then everyone will think I'm cool."

b) Translate the following sentence in to predicate logic, capturing as much of the meaning as possible: "Every habitable planet is orbited by a moon."

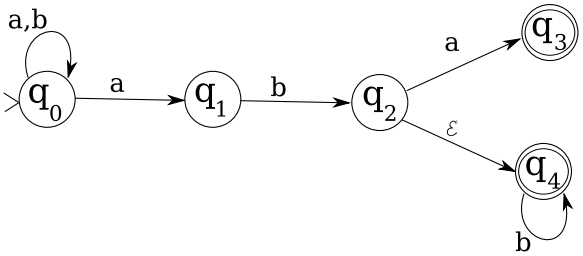
c) The statement "Everyone fears something" is ambiguous since it could mean either $\forall x\, \exists y\, F(x, y)$ or $\exists y\, \forall x\, F(x, Y)$, where the meaning of the predicate $F(x, y)$ is "x fears y." Give **unambiguous** English translations of the two statements.

4. Translate the following argument into logic, and either give a formal proof that the argument is valid or explain why it is not valid: "If Jack has a day off from work, it's raining. If the sun is shining, then it's not raining. Jack has a day off from work, so the sun is not shining."

5. Give an example of a set A such that $A \cap \mathcal{P}(A) \neq \emptyset$. Show both A and $\mathcal{P}(A)$. (Recall that $\mathcal{P}(A)$ is the power set of A.)

6. Draw a DFA that accepts the language $L = \{w \in \{a, b\}^* \mid \text{every } a \text{ in } w \text{ is immediately followed by } bb\}$

7. Use the NFA-to-DFA conversion algorithm, in which states in the DFA correspond to sets of states in the NFA, to construct a DFA that accepts the same language as the following NFA:



8. Identify the language generated by the regular expression $(ab|ba|aa|bb)^*(a|b)$. Explain your reasoning.

9. **Two** of the following languages are contex free. Give context-free grammars for the languages that are context free. You do not have to explain your grammars.

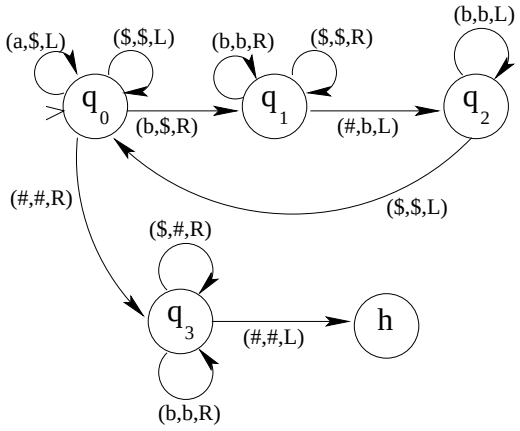
$\{a^n b^m a^n b^m \mid n, m \in \mathbb{N}\}$ $\{a^n b^m a^m b^n \mid n, m \in \mathbb{N}\}$ $\{a^n b a^m b a^{n+m} \mid n, m \in \mathbb{N}\}$

10. **Prove** the following statement: Suppose that A, B, and C are sets and that $A \subseteq B$ and $A \subseteq C$. Then $A \subseteq B \cap C$.

11. Shown below is a grammar for $L_1 = \{a^{2^n} \mid n \in \mathbb{N}\}$ that we looked at in class. Write a similar grammar that generates the language $L_2 = \{a^{3^n+1} \mid n \in \mathbb{N}\}$. Explain how your grammar works.

$$\begin{aligned} S &\longrightarrow XTaY \\ T &\longrightarrow DT \\ T &\longrightarrow \varepsilon \\ Da &\longrightarrow aaD \\ DY &\longrightarrow Y \\ Xa &\longrightarrow aX \\ XY &\longrightarrow \varepsilon \end{aligned}$$

12. The Turing Machine that is shown at the right is designed to process strings over the alphabet $\{a, b\}$. Assume that we always start this Turing Machine on the **rightmost** character of its input string.



a) What does this machine do when it is started on an empty string?

b) What does this machine output when it is started with input aba?

c) Describe the computation and output of this machine when it is started with any input string $w \in \{a, b\}^*$.

13. Prove that for any language L, if $\varepsilon \in L$, then $L \subseteq LL$.

14. In this course, we have looked at various kinds of langauges and abstract machines. Write an essay discussing the different kinds of languages, the different kinds of machines, the relationships among them, and what you have learned from all this about the nature of computation.