Please write your name at the top of this page and nowhere else.

In addition to this cover page, there should be five problems, A–E, spread over five pages.

No notes, books, calculators, computers, etc. are allowed.

Feel free to ask clarifying questions. If a problem is unclear and you cannot obtain clarification, then write your interpretation of the problem, so that I can evaluate your solution relative to your interpretation. You might be penalized, if your interpretation makes the problem much easier than it should be. Certainly you should never interpret a problem in a way that renders it trivial.

Except where otherwise noted, all problems require explanation. Incorrect answers with good work often earn partial credit. Correct answers without justification rarely earn full credit. Write as if your audience is a typical classmate — not a professor. In doing so, you (hopefully) show enough detail, that I can evaluate whether you understand your arguments.

You may cite material (definitions, algorithms, theorems, etc.) that we have defined or proved in class, in the assigned textbook readings, or in the assigned homework. You do not need to re-define or re-prove any of that material. You may not cite other material without developing it first.

You have 70 minutes. Good luck. :)

Here are eight examples of floating-point literals — that is, strings that represent floating-point constants — in a hypothetical programming language.

`-3.14159` `21.0` `-7` `02.72` `2.45733e4` `-17e1` `-0.14E-6` `0.999989`

In this context, `e` and `E` mean "times 10 to the power". For example, `5.1e2` and `5.100E02` both represent 510. The power is always an integer. The decimal point is always preceded and followed by a digit. The `e` and `E` are always preceded by a digit and followed by either a digit or `-`. The `-` is always followed by a digit. For comparison, here are eight strings that are not valid floating-point literals.

`-.14159` `2.` `-` ϵ `2.e4` `-e1` `-0.14E-` `--0.999989`

A.A. What is the underlying alphabet for this programming language's floating-point literals?

A.B. Write a regular expression that matches this language's floating-point literals. Use textbook syntax, not Python syntax. For clarity, please employ a regular sub-expression D that matches the ten digits $\texttt{0}, \texttt{1}, \texttt{2}, \ldots, \texttt{9}$.

B. Over $\Sigma = \{a, b, c\}$, let A be the set of all strings w such that the number of as in w, times the number of bs in w, equals the number of cs in w. Is A regular? Prove your answer.

C. Let $A = \{(01)^m : m \geq 0\} \subseteq \{0,1\}^*$. Is A regular? Prove your answer.

D. Let $A = \{a^i b^j c^k : i \geq 0, j \geq 0, k \geq 0, \text{ and } (i \neq j \text{ or } j \neq k)\} \subseteq \{a, b, c\}^*$. In English, describe a PDA for A. Your description should be precise enough, that a classmate could then draw the PDA from it. If you doubt so, then feel free to draw the PDA yourself.

E. On each TRUE-FALSE question below, there are four valid answers. If the correct answer is TRUE, then TRUE earns 3 points, TRUISH earns 2 points, FALSISH earns 1 point, and FALSE earns 0 points. If the correct answer is FALSE, then these point values are of course reversed. Do not write just T or F; write your answer completely and clearly. No explanation is needed!

E.A. If A is regular and $B \subseteq A$, then B is regular.

E.B. For any context-free language A, there exists a PDA whose language is A.

E.C. For any regular language A, there exists a CFG whose language is A.

E.D. If A is regular and p is a pumping length for A, then $p+1$ must also be a pumping length for A.