Notes, book, etc. are not allowed.

Except where otherwise noted, you must justify your answers. Correct answers with no justification may receive little credit. Incorrect or incomplete answers that display insight often receive partial credit.

You may cite material (definitions, theorems, algorithms, etc.) discussed in class, assigned homework, or the assigned textbook sections. If you wish to use other material, then you must develop it first.

It is understood that efficient, concise solutions are usually favored over inefficient or verbose solutions, and hence may earn more points.

If you feel that a problem is ambiguously worded, then ask for clarification. If the problem is still unclear, then explain your interpretation in your solution. Never interpret a problem in a way that renders it trivial.

You have 70 minutes. Good luck. :)

A. Many text editor applications, which programmers use to write code, have search-and-replace functions capable of handling regular expressions. The details vary from editor to editor, so let's suppose that your editor uses textbook regular expressions, where the only valid operators are $\cup$, $\circ$, and *. Parentheses are used to indicate precedence, so `\(` and `\)` are used encode literal parentheses to be matched. While revising your latest programming project, you decide that it would be helpful to find all chunks of code that look like these examples:

```
veshInitializeStyle(&
shaInitialize(&
mesh3DInitializeFile(&
```

More precisely, you want to find all strings that contain one or more alpha-numeric characters, followed by the string `Initialize`, followed by zero or more alpha-numeric characters, followed by one opening parenthesis `(`, followed by one ampersand `&`.

Write a textbook regular expression that matches all such strings (and no other strings).

B. In this problem, I ask you to start — but not finish! — proofs that three languages are not regular. Each proof begins, "Assume for the sake of contradiction that the language is regular. Let p be the pumping length guaranteed by the pumping lemma. Let $w \ldots$." Your job is to give a value of w that could be used to complete the proof. Remember that simple correct answers are preferred over complicated ones. Clearly mark your w. You do not need to write or explain any of the proof before or after defining w.

B.A. Let $A = \{xyx : |x| \geq 1, |y| \geq 1\}$ over $\Sigma = \{0, 1\}$.

B.B. For any string $x \in \{0, 1\}^*$, let $\overline{x}$ be the bitwise negation of x. For example, $\overline{1011} = 0100$. Let $B = \{x\overline{x} : x \in \{0, 1\}^*\}$.

B.C. Let $\Sigma = \{0, 1, =, \wedge\}$. Let C be the set of strings of the form $x \wedge y = z$, where $x, y, z \in \{0, 1\}^*$ and the equation is a valid equation of binary numerals with $\wedge$ meaning "to the power of".

C. In this problem, the alphabet $\Sigma = \{$`a`, `b`, ..., `z`, `,`, `(`, `)`$\}$ consists of 29 characters. We're trying to describe the syntax of Python-like function calls such as `salary(x,age,edu)` and `thunk()`. Each function call begins with the function name. Then comes a pair of parentheses enclosing zero or more arguments separated by commas. Identifiers (names of functions, variables, constants, etc.) consist of one or more letters — not commas or parentheses. Function calls can be nested, as in this more complicated example: `join(xyz,new(a,length(b)),end)`.

Write a CFG for function calls as just described. Consider adding comments to your CFG, to clarify the intent of any confusing or complicated parts. For example, for each CFG variable, it would be helpful to either give it a meaningful name or describe its meaning.

D. On these TRUE-FALSE questions 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.

D.A. Let Σ be any alphabet, $A \subseteq \Sigma^*$ any language over Σ, and $b \geq 0$. Let $A_{\leq b} = \{w : w \in A, |w| \leq b\}$. Then $A_{\leq b}$ must be regular.

D.B. Let A_1 be a language over Σ_1 and A_2 a language over Σ_2. Then A_1 and A_2 are both languages over $\Sigma_1 \cup \Sigma_2$.

D.C. If A is a context-free language, then A^c must also be context-free.

D.D. Let $\Sigma = \{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \ldots, \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}\}$ be the alphabet consisting of all sixteen possible columns of four 0s and 1s. Let $A \subseteq \Sigma^*$ be the set of all strings corresponding to valid binary addition problems, where the first three rows are added to produce the final row. (We did a homework problem that was similar but smaller.) Then A is regular.

D.E. For any PDA N, there exists a PDA P such that $L(P) = L(N)$, P has no transitions where it neither pops nor pushes, and P has no transitions where it both pops and pushes.