This exam consists of four problems spread over six pages (including this page).

Notes, book, etc. are not allowed.

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

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. Give an example of two CFLs A and B (over the same alphabet), such that the intersection $A \cap B$ is not a CFL. Justify all parts of your answer. (Hint: Set things up so that $A \cap B$ is a language that we already know not to be a CFL.)

B. You work for a company that uses Python for many in-house computing tasks. Your team uses a lot of old code, all of which is documented as in the example at left: Each function is preceded by a comment string enclosed in triple quotation marks. On the other hand, all of the new code written by your team is documented as in the example at right: a "docstring" immediately after the function header, enclosed in triple quotation marks.

```
"""Returns the intersection of two
sets. Inputs and output are tuples.
Output is ordered to match order
of X."""
def intersection(X, Y):
    res = []
    for x in X:
        if x in Y:
            res.append(x)
    return tuple(res)
```

```
def union(X, Y):
    """Returns the union of two sets.
    Inputs and output are tuples.
    Output is ordered to match order
    of X."""
    res = list(Y)
    for x in X:
        if not x in Y:
            res.append(x)
    return tuple(res)
```

Your boss, Ms. Squiddlesby-Octothorpe, wants you to create a reference manual for the team. It should give the header and documentation for each function — for example:

```
union(X, Y)
Returns the union of two sets. Inputs and output are tuples. Output is ordered
to match order of X.
```

You quickly realize that you can scrape this information from the team's Python files using regular expressions. In fact, just a few lines of Python code is enough to generate the following list of tuples of strings, from a Python file containing the two functions above. Once you have these tuples, writing additional code to format them into a reference manual will not be difficult.

```
[('Returns the intersection of two sets. Inputs and output are tuples. Output
is ordered to match order of X.', 'intersection(X, Y)', '', ''), ('', '',
'union(X, Y)', 'Returns the union of two sets. Inputs and output are tuples.
Output is ordered to match order of X.')]
```

So how do you get the tuples? Answer on the following sheet. For simplicity, you may assume that documentation never includes the character ". (It's okay if you don't remember the exact function names and Python regular expression syntax. It's okay if you choose to extract the information in a slightly different way. But give the regular expression and surrounding code as well as you can, and explain in English how it is all supposed to work.)

This page offers more than enough room to answer Problem B.

C. Let A be the set of all strings s over the alphabet $\{a, b, c, d\}$ such that the number of as in s equals the number of bs in s times the number of cs in s. If A is regular, then draw a DFA or NFA for it. If A is not regular, then prove so.

D. In each part of this problem, there are three valid answers: TRUE, FALSE, and PUNT. If you answer PUNT, then you get half credit. Otherwise, you get full credit for answering correctly and no credit for answering incorrectly. Justification is not required on this problem.

1. If A is any CFL over Σ, then the complement $\Sigma^* - A$ is also a CFL.

2. If A is any language over Σ and the complement $\Sigma^* - A$ is finite, then A is regular.

3. If an NFA is run on input w, and one branch of computation accepts, and one branch of computation rejects, then the NFA accepts w.

4. It is impossible for a CFG in Chomsky normal form to derive the empty string ϵ.

5. The transition function of an NFA has the form $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$.

6. The reverse of any CFL is also a CFL.