Compilers [Fall 2017]
Test I

NAME: _____

Instructions:

1) This test is 7 pages in length.

2) You have 75 minutes to complete and turn in this test.

3) Essays and short-answer questions will be graded on how clearly you've communicated the necessary ideas. Write in complete English sentences.

4) This test is closed books, notes, papers, friends, phones, neighbors, smartwatches, etc.

5) Use the backs of pages in this test packet for scratch work. If you write more than a final answer in the area next to a question, circle your final answer.

6) Write and sign the following: "I pledge my Honor that I have not cheated, and will not cheat, on this test."

_____

_____

Signed: _____

1. [10 points] [Short essay, 3-5 sentences]
With regards to CFGs, what is a derivation? Include an example in your explanation.

2. [10 points]
Write a flex rule (i.e., regular expression) for tokenizing floating-point literals, which are defined as follows:

- A float may optionally begin with a plus or minus sign.
- A float may optionally end with an exponent portion. Exponent portions of floats must begin with the character e or E, then may contain a plus or minus sign, and then must contain a nonempty sequence of digits.
- A float may contain any natural number of digits but must contain at least one digit.
- A float must contain at least one digit before the exponent portion, if one is present.
- A float may optionally contain a decimal point. Any natural number of digits may appear before a decimal point, but at least one digit must appear after a decimal point. If the float contains both a decimal point and an exponent portion, then at least one digit must appear between the decimal point and the exponent portion.

Keep your response simple enough that it cannot be simplified in any significant ways.

3. [15 points]
Let R be the RE (1|01|001)*(ε|0|00).
(a) In English, succinctly describe the strings R matches. [1 sentence]

(b) Define a DFA recognizing exactly those strings over {0,1} matched by R. Show that your DFA is minimized.

4. [15 points]
Define an algorithm that inputs RE_1 and RE_2 (which both use the same alphabet Σ) and outputs a DFA accepting exactly those strings matched by both RE_1 and RE_2.

5. [25 points]

G is:

```
0   S -> A$
1   A -> AxB
2   A -> ε
3   A -> y
4   B -> By
5   B -> ε
```

a) Draw an LALR(1) parse table for G.

b) Show an LALR(1) parse trace for input xxy$ according to G. If the trace ever gets stuck due to a conflict or parse error, just note the conflict or error at the point it occurs and stop the trace at that point.

6. [25 points]
Consider the following grammar for multiplying numbers (n).

```
S-> E$
E-> n | E*E
```

Let's add properly balanced parentheses to these multiplication expressions. For example, we should allow the following program to be considered valid:

```
((n)*((n)(n)))(((n*n*n)))$
```

a) Write a grammar that exactly matches multiplication expressions with balanced parentheses.

b) If your grammar from Part (a) is left-recursive, remove the left recursion, and if your grammar can be left factored, do so. For whatever grammar you end up with, after these at-most-2 transformations, show the LL(1) parse table.

Undergraduates stop here. The remaining problem is for graduate students.

7. [15 points]

G' is:

```
0    S -> A$
1    A -> B A
2    A -> ε
3    B -> ε
4    B -> x
```

Draw an LR(2) parse table for G'.