Compilers [Fall 2016]
Test II

NAME: ____

Instructions:

1) This test is 7 pages in length.

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

3) Prose-response questions include a guideline for how much write. Respond in complete English sentences. Essays should be well organized and readable.

4) This test is closed books, notes, papers, friends, neighbors, 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. [5 points]
What is the vocabulary word the class had trouble remembering during our last meeting? What does that word mean? [1-2 sentences]

2. [5 points]
What is GLR parsing? At a very high level, how do GLR parsers work? [2-3 sentences]

3. [10 points]
Draw a minimum-state DFA accepting exactly those strings over {*o*,*n*} that don't contain *nono* as a substring. If no such DFA exists, write "impossible".

4. [15 points]

G is: `0 S -> N$   1 N -> NI   2 N -> q   3 I -> ε   4 N -> N   5 I -> q`

(a) Draw an SLR parse table for G.

(b) Show an SLR parse trace for the input string qq$ according to G. If the trace ever reaches a state of conflict, write "conflict" at that point and stop the trace.

5. [65 points] [Essay]
Let's consider adding *case* (also known as *switch*) expressions to DJ. The format is

$$\text{case(e)}\ \{n_1\ \{el_1\}\ \ n_2\ \{el_2\}\ \ldots\ n_i\ \{el_i\}\ \ \text{default}\ \{el_d\}\}$$

where e is an expression, all the n's are natural numbers, all the el's are expression lists, and i is a positive integer. All the natural numbers $n_1..n_i$ must be distinct.

Operationally, expression e gets evaluated to a natural number n. If $n=n_1$ then expression list el_1 gets executed and its result becomes the result of the entire *case* expression; if $n=n_2$ then expression list el_2 gets executed and its result becomes the result of the entire *case* expression; etc.; and if n equals none of $n_1..n_i$, then expression list el_d gets executed and its result becomes the result of the entire *case* expression. Note that DJ *case* expressions don't "fall through" from one case to another, as occurs in C/Java *switch* statements. However, like the typical use of C/Java *switch* statements, DJ *case* expressions get be compiled into code that uses jump instructions to execute more efficiently (generally speaking) than equivalent *if-then-else* expressions. The price for this time-efficiency may be increased code-memory usage, but for this problem we'll assume arbitrarily much code memory is available.

Suppose you have the source code for a complete dj2dism compiler, implemented as discussed in class. **How would you modify dj2dism, to add support for DJ *case* expressions?** Use pseudocode but be specific. The next two pages are blank, in case you need more space.

(Additional space for Problem 5)

(Additional space for Problem 5)

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

6. [15 points] [Essay]
How do LL(*) parsers work? Analyze and explain their worst-case running time and how it compares to that of LL(2) parsers.