Homework #4
Semantic Analysis

#1. (1 point) **Consider the attribute grammar**:

N1 --> N2 c	(i) N1.cnum = N2.cnum + 1 (ii) N1.dnum = N2.dnum
2. N1 --> N2 d	(i) N1.cnum = N2.cnum (ii) N1.dnum = N2.dnum + 1
3. N --> d	(i) N.cnum = 0 (ii)N.dnum = 1
4. N --> c	(i) N.cnum = 1 (ii)N.dnum = 0

where the numbers on the N are just to distinguish them.

a) **Show that the underlying grammar is or is not LL(1)**

b) **Show that the underlying grammar is or is not SLR(1) (That is, can we create the states and tables as in Homework #4)**

c) **Draw a parse tree and compute attributes for cddc**

d) **What do the attributes do?**

#2. (1 point) Show and AST and quads for

Rich = (assets > million) and NOT InDebt

#3. (1 point) Translate the statement: WHILE (1 < P) AND (P < 3) DO P := P + Q

into:

(a) an abstract syntax tree

:

(b) quadruples

(c) triples

(d) indirect triples

#4. Consider the following attribute grammar for a program consisting of a single copy rule:

P → **{** S **}**	S.*Live* = S.*Use*
S → A **;**	S.*Use* = A.*Use* S.*Def* = A.*Def* A.*Live* = S.*Live*
A → V **=** E	E.*Use* = {**LexVal**(variables in E)} V.*Def* = {**LexVal**(V)} A.*Def* = V.*Def* A.*Use* = E.*Use* **-** V.*Def* (**-** is set difference) V.*Live* = A.*Live* **-** V.*Def* (**-** is set difference) E.*Live* **=** E.*Use*

where E can be any valid expression such as *a* + *b*

(a) **(1/3 Point)** List the synthesized and inherited attributes:

(b) **(1/3 Points)** Parse and evaluate attribute for the program:

(c) **(1/3 Point)** What might these attributes be used for?

#5. (Choose all that apply) (1 point) Evaluating semantic attributes is:

a) tractable
b) intractable
c) decidable
d) undecidable
e) NP-Complete