CSE 6341, Written Assignment 1
Due Thursday, September 5, 11:59 pm (8 points)

Your submissions should be uploaded via Carmen. Create your answers using a text editor and upload the file (e.g., plain text, Word, PDF). Alternatively, you can write your answers by hand and take a photo (or scan), but please ensure that (1) your handwriting is *clear and legible*, and (2) your photo or scan has *high resolution*, to allow the grader to read and understand your submission.

Q1 (2 points): In class we discussed a grammar for a simple language of expressions (slide 12 of the notes). We also showed the leftmost derivation of string **int x = 1; y = x + 2;**
Show the *rightmost* derivation of string **int y = z + x; int x = z;** with this grammar.

Q2 (3 points; 2 parts): Consider the grammar from Q1, with the following modifications:
<stmt> ::= <varDecl> **;** | <varDecl> **=** <expr> **;** | **ident =** <expr> **;**
<varDecl> ::= **int ident** | **float ident**
<expr> ::= **intconst** | **floatconst** | **ident** | <expr> **+** <expr>

Part 1. Consider some parse tree for
int b = x + y + z; float c = b;
How many nodes are in this tree? How many of these nodes are leaf nodes? You do *not* have to show a parse tree; just state the number of nodes and the number of leaf nodes.

Part 2. This grammar is ambiguous. Explain whether/how this affects your answer for Part 1.

Q3 (3 points; 2 parts): For the program from Q2, consider the AST from Project 1. There are 11 non-abstract classes in Java package *ast* (excluding helper class Location). AST nodes in Project 1 are Java objects that are instances of some of these non-abstract classes (recall that abstract classes do not have instances).

Part 1. For each of these 11 classes, show how many instances (i.e., objects) of this class exist in the AST for this program.

Part 2. Show the subtree of the AST corresponding to expression **x + y + z**. When showing each AST node, label it with the corresponding Java class name (for the non-abstract class of AST node). Based on the shape of this AST subtree, discuss whether the parser provided to you in Project 1 uses left-associativity or right-associativity for operator **+**.