Reading

2.	Friday 21 Jan:	Sethi: Chapter 1
3.	Monday 24 Jan:	Sethi: Sections 2.0, 2.1, 2.2, 2.3
4.	Wednesday 26 Jan:	Sethi: Sections 2.4, 2.5, 2.6
5.	Friday 28 Jan:	Davie[1]: Chapter 1, Sections 2.1–2.4; Gentle Intro: 3.2, 4.3
6.	Monday 31 Jan:	Davie: Sections 2.14.3, 2.12, 2.3, 3.5; Gentle Intro: 3.1

Exercises

Due date: Tuesday 1 February. Please leave your paper in the Taylor Hall homework box (it's located in the breezeway between 2.132 and 2.136) **by 4pm**.

The homework box is being shared by several classes, so it's important for you to ***staple*** your paper's pages securely, and to ***use the cover sheet*** supplied with this assignment.

Legibility and organization count. The graders' time is limited, and it's up to you to make your answers clear and convincing. An answer that's not clearly correct will be considered incorrect.

To help the graders, please arrange your answers in the same order as the exercises. And don't forget to write your name(s) on the cover sheet.

Numbers in [brackets] are point values; the point total for this assignment is **195**. Although point totals vary from one assignment to the next for convenience in grading, all assignments (except the lowest) contribute equally to your course grade.

1. [15] Rewrite the expressions **a**–**e** below in *prefix* notation. Assume the following (customary) precedences and associativities for the operators:

^	3	right
*	2	left
/	2	left
+	1	left
-	1	left

and don't assume that any of the operators are commutative.

A simple way to proceed is to perform the rewriting process in three steps:

1. Parenthesize the expression fully, using the rules of precedence and associativity.
2. Within each parenthesized subexpression, move the operator to the left of its left operand.
3. Remove the parentheses.

You'll notice that in the resulting expression, the order of the operands is the same as in the original— only the operators change position.

a. a * b / c - d **b.** a * (b + c / d)
c. a * b + c / d **d.** a ^ b ^ c * d
e. a * b * c ^ d

2. [15] Rewrite the expressions of Exercise 1 in *postfix* notation. Use the same assumptions as in Exercise 1.

3. [15] Construct abstract syntax trees for the expressions of Exercise 1. Use the same assumptions as in Exercise 1. A useful first step is to parenthesize each expression fully before drawing its tree.

4. [15] For the following expression

a • b • c Δ d Δ e • f Δ g Δ h

assume • has precedence 1 and is right-associative, and Δ has precedence 2 and is left-associative.

a. Rewrite the expression in prefix notation.
b. Rewrite the expression in postfix notation.
c. Draw the expression's abstract syntax tree.

[1] "Davie" refers to *Readings from "Intro to Functional Programming Systems Using Haskell"*.

5. [10] Sethi, Exercise 2.4c.

6. [10] Repeat Exercise 5, expressing the grammar in EBNF.

7. [10] Repeat Exercise 6, expressing the grammar in syntax charts.

8. [20] The C programming language supports multiple formats for numeric literals.

- Octal literals are numerals that begin with the digit ‘0’ and contain only digits in {0, …, 7}.
- Hexadecimal literals start with the characters “0x” or “0X” and contain digits in {0, …, 9} ∪ {A, …, F}.
- Decimal literals are numerals that start with any digit other than ‘0’ and contain digits in {0, …, 9}.

Write a context-free grammar that describes the octal, hexadecimal, and decimal literals as defined above.

9. [10] Sethi, Exercise 2.5.

10. [15] Sethi, 2.9 **a**, **c**, **e**. Your descriptions should be in English.

11. [15] For each of the following strings, draw a parse tree with respect to the grammar for arithmetic expressions in Sethi’s Fig. 2.6 (page 42):

a. `x + 1 - 2`
b. `x + (1 - 2)`
c. `x * y + z`
d. `x * (y + z)`
e. `x + y * z`

12. [15] Draw an abstract syntax tree for each of the expressions in Exercise 11.

13. [15] Extend the grammar of Sethi’s Fig. 2.6 (page 42) to include an exponentiation operator (^). Define the new grammar so that (^) has a higher precedence than (*) and is *right*-associative.

14. [15] Consider the following grammar:

Start ::= *Benny* • *Joon* | *Joon*
Benny ::= *Benny* ▲ *Benny* | *Joon*
Joon ::= a | b | c

a. Demonstrate, using one or more sample strings and their parse trees, that the grammar is ambiguous.

b. Change the grammar to an unambiguous one that describes the same language and makes the ▲ operator right-associative.

CS 345 Richards

Assignment 1

Authors:

1. ______________________

2. ______________________

3. ______________________

4. ______________________