CPSC 499 # Functional Data Structures *UNBC*

Winter 2006

Prerequisites: A C$^-$ in CPSC 281 and CPSC 370 and all of their pre-requisites, or permission of the instructor.

Instructor: Dr. David Casperson

Office: Lib 5-471

Telephone: 960-6672

e-mail: `casper@unbc.ca`

web address `http://web.unbc.ca/~casper/Courses/2006-499.html`

Course Goals are:

- to understand the functional programming idiom;
- to understand why functional programming requires new data structures;
- to understand amortized complexity;
- to understand persistent and ephemeral data-structures;
- to understand strict versus lazy evaluation;
- to understand complexity analysis in the presence of laziness, and the role of laziness in constructing amortized-time persistent data structures.

Required Text: *Purely Functional Data Structures* by Chris Okasaki.

Suggested references: *Elements of ML Programming* (*2nd edition*) by Jeffrey D. Ullman.
ML for the Working Programmer by L. C. Paulson.

Web references: Available by following links from `http://www.smlnj.org/`
Programming in Standard ML (*2nd edition*) by Robert Harper.
Notes on Programming in Standard ML of New Jersey by Riccardo Pucella.

Grading: (subject to revision)

Homework	:	15%
Exam 1	:	25%
Exam 2	:	25%
(Final) Exam 3	:	35%

Class times:

Room,	Day,	and Time
8-161	T	13:00–14:20
8-161	F	10:00–11:20

Some Dates

First class	Tue	2006-01-03
Midterm I	Tue	2006-02-07
Winter Break	2006-02-13—2006-02-17	
Midterm II	Tue	2006-03-07
Last drop day	Wed	2006-02-22
Course evaluation	Fri	2006-03-31
Last class	Tue	2006-04-04
Final Exam	2006-04-07—2006-04-22	
Holiday	Fri	2006-04-14
Holiday	Mon	2006-04-17

Topics will be chosen from among the following.

- An introduction to functional programming and Standard ML.
- Static versus dynamic typing. Polymorphic typing. Type inference.
- Strict versus lazy evaluation.
- "Pure" versus "impure" functional programming.
- Functions and partial functions.
- The notion of Currying.
- Forms of recursion, accumulators, continuations, cata-morphisms.
- Space and time complexity for functional programs and data structures.
- Persistant and ephemeral data structures.
- Amortized complexity and analysis of lazy data structures.
- Imperative implementation of functional data structures.
- Strategies for re-adjusting time complexity.