Programming Languages

https://web.unbc.ca/~casper/Semesters/2021-05F/320.php

Accommodations:

If there is any student in this course who, because of a disability, may have a need for special academic accommodations, discuss this with me, or contact the Access Resource Centre located in Teaching & Learning 10-1048.

Prerequisites: A C$^-$ or better in CPSC 242, and CPSC 200; or permission of instructor.

Who, Where, When?

ROOMS	Lectures are in 5-176
HOURS	17:30–18:20 **M W F**
E-MAIL	David.Casperson@unbc.ca
INSTRUCTOR	David Casperson
OFFICE	T&LC 10-2080
TELEPHONE	(250)960-6672

Text:

Required *Programming Languages*. by R. W. Sebesta. (11/12)$^{\text{th}}$ Edition.

Grading and Dates:

First Class		Wed, Sep 8
Thanksgiving		Mon, Oct 11
Exam 1	**25%**	Fri, Oct 15
last drop		Thu, Oct 29
Remembrance day		Thu, Nov 11
Exam 2	**25%**	Mon, Nov 22
Language Presentation	**20%**	???
Last Class		Mon, Dec 06
Participaton	**10%**	
Homework	**20%**	

Why?

Programming Language popularity and availability changes constantly, and most programmers learn multiple languages and paradigms. This course introduces introduces general ideas that underly programming languages and their design and description, giving a framework for reasoning about, learning, and designing computer languages.

What? Topics chosen from (*not necessarily in the order listed*):

- Syntax: Lexical analysis — tokens, Concrete syntax, Abstract syntax, Grammar descriptions, (E)BNF, Ambiguity, derivation trees.
- Formal semantics: Operational, Axiomatic, & Denotational Semantics. Small and big step semantics.
- Common semantics: bindings, scope, environments, allocation, lifetime. Lexical versus dynamic binding.
- Calling mechanisms. Lazy evaluation. Call by -value, -reference, -copy and return, -name, -textual substitution.
- Tyeps and data types. Mathematical models. Hindley-Milner type systems and HASKELL. Polymorphism. Type inference. Type equivalence and type checking.
- Control Structures. Selection, looping, and non-local flow. Procedures and Environments. Recursion. Parameter-passing mechanisms. Exception handling. Continuations.
- Programming in the large. Modules and packages. Information hiding, data abstraction. Object-based and object-oriented programming.
- Design principles. Simplicity, abstraction, orthogonality, reliability, ...