

Course #:	CSCI-C 311
Course Title:	Structure of Programming Languages
Course Type:	Required core
Prerequisites:	C243 Data Structure, C335 Computer Structures
Credits:	3
Text Book:	Programming Language Pragmatics, 2nd edition, by M. L. Scott. Morgan Kaufmann, 2006.
References:	Handouts Elisp manual List of Emacs keyboard commands Essentials of Programming Languages, by Friedman, Haynes, & Wand
Current Catalog Description:	Design and implementation of programming languages: syntax; semantics; comparison of programming paradigms such as imperative, functional, logic, and object oriented. Implementation of concepts such as binding, scope, looping, branching, subprograms and parameter passing, tasks and concurrency, heap management, exception handling, templates, inheritance, overloading.
Course Goals	The student who completes this course: <ol style="list-style-type: none"> 1. Will understand language translations as related to compilers and interpreters. 2. Will understand the difference between procedural, functional, and declarative languages. (C, Lisp, Prolog). 3. Will be proficient in reading and writing grammars (BNF) 4. Will understand the basic concepts behind language constructs such as loops, arrays, functions, pass by value, pass by address, activation stack, variable scope, etc. 5. Will understand the different passes of the compiler and processes such as lexical analysis, parsing and code generation. 6. Will be given the opportunity to develop an interpreter for a small programming construct (i.e. function call, arrays, or the notion of safe pointers and garbage collection.)
Major Topics Covered in the Course	<ol style="list-style-type: none"> 1. Compiled languages vs. interpreted languages. 2. Introduction to Lisp, the Emacs editor and interpreter for Lisp 3. Major categories of programming languages 4. First class objects, higher order procedures. 5. Recursion, tail recursion elimination, deep recursion, dynamic programming 6. Program translation phases.

	<ol style="list-style-type: none"> 7. Formal grammars, Backus-Naur notation, finite states machines 8. Scope, dynamic scope, lexical address 9. Interpreters for little languages 10. Parsing and parse trees 11. Parameter passing methods 12. Compiler compilers 13. Practical comparison of existing programming languages 																		
Laboratory projects (specify number of weeks on each)	<p>2 laboratories, each of them half a week, 1.5 hours</p> <p>Lab 1: familiarization with the Emacs editor and interpreter for Lisp, writing Lisp code in Emacs, evaluation modes and commands</p> <p>Lab 2: an independent search for and comparison of available compiler compiler or parser generator software</p>																		
Estimate Curriculum Category Content (Semester hours)	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Area</th> <th>Core</th> <th>Advanced</th> </tr> </thead> <tbody> <tr> <td>Algorithms</td> <td>13</td> <td>4</td> </tr> <tr> <td>Software Design</td> <td>5</td> <td></td> </tr> <tr> <td>Comp. Arch.</td> <td></td> <td></td> </tr> <tr> <td>Data Structures</td> <td>5</td> <td></td> </tr> <tr> <td>Prog. Languages</td> <td>20</td> <td>5</td> </tr> </tbody> </table> <p>Additional hours may be dedicated to curriculum categories not listed above. For example explanation of concepts and theories. Discussion of social and ethical issues, discussion of inter personal relationships and working within groups.</p>	Area	Core	Advanced	Algorithms	13	4	Software Design	5		Comp. Arch.			Data Structures	5		Prog. Languages	20	5
Area	Core	Advanced																	
Algorithms	13	4																	
Software Design	5																		
Comp. Arch.																			
Data Structures	5																		
Prog. Languages	20	5																	
Oral and Written Communications	Every student is required to submit at least __1__ written reports of typically __3 to 15__ pages. Oral presentations of typically __15__ minute's duration.																		
Social and Ethical Issues	Not a course objective.																		
Theoretical Content	<ul style="list-style-type: none"> • Classification of programming languages, 1 h • The difference between compiled and interpreted languages, 30 minutes • Formal grammars, 2 h • Finite states machines, 30 minutes • First class objects, first class functions, 2 h • Scope, dynamic scope, 3h • Program translation phases, 2 h 																		
Problem Analysis	Not a course objective.																		
Solution Design	Not a course objective.																		
Prepared By	Vrajitoru, Hakimzadeh																		