Programming Languages	Code & No:	CS 270
	Credits:	3 (3,0,1)
	Pre-requisite:	CS 210
	Co-requisite:	None
	Level:	6

Course Description:

This course describes a set of formal mathematical tools for defining and implementing the semantics of a language and demonstrates them in the context of important real-world programming languages, with emphasis on theoretical properties of type systems. Major topics include: lexical and syntax analysis, bane binding, type checking and scopes, data types, expressions, flow control, and subprograms.

Course Aims:

1. Acquire the fundamental concepts of programming languages and techniques to discuss and compare features of several popular programming paradigms such as imperative, object oriented, functional, and logic programming.
2. Understand how to examine modern programming languages and features: abstract data and control structures, procedures, parameter passing mechanisms, block structuring and scope rules, input/output, and storage management
3. Learn the vocabulary of programming language design, syntax, and semantics, develop an understanding of how programming languages differ and learn how to describe concrete syntax and how that syntax drives the structure of translation programs

Student Outcomes (SOs):

☒(a) An ability to apply knowledge of computing and mathematics appropriate to the program's student outcomes and to the discipline

☒(b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution

☒(c) An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs

☐(d) An ability to function effectively on teams to accomplish a common goal

☐(e) An understanding of professional, ethical, legal, security and social issues and responsibilities

☐(f) An ability to communicate effectively with a range of audiences

☐(g) An ability to analyze the local and global impact of computing on individuals, organizations, and society

☐(h) Recognition of the need for and an ability to engage in continuing professional development

☒(i) An ability to use current techniques, skills, and tools necessary for computing practice.

☒(j) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices. [CS]

☐(k) An ability to apply design and development principles in the construction of software systems of varying complexity. [CS]

☐(j) An ability to use and apply current technical concepts and practices in the core information technologies of human computer interaction, information management, programming, networking, and web systems and technologies. [IT]

☐(k) An ability to identify and analyze user needs and take them into account in the selection, creation, evaluation, and administration of computer-based systems. [IT]

☐(l) An ability to effectively integrate IT-based solutions into the user environment. [IT]

☐(m) An understanding of best practices and standards and their application. [IT]

☐(n) An ability to assist in the creation of an effective project plan. [IT]

Course Learning Outcomes (CLOs):

1. Describe the evolution of modern programming language
2. Identify the basic aspects of various programming paradigms.
3. Demonstrate facility with BNF for specifying programming language syntax
4. Specify various control structures using operational semantics
5. Show understanding of issues involving variables: data types, binding, strong typing, and scope
6. Describe various subprogram parameter passing methods

SOs and CLOs Mapping:

CLO/SO	a	b	c	d	e	f	g	h	i	j	k	l	m	n
CLO1			√											
CLO2		√												
CLO3	√													
CLO4									√					

CLO5										√				
CLO6										√				

No.	Topics	Weeks	Teaching hours
1	Reasons for studying concepts of programming languages, Language evaluation criteria, Language Categories	1	3
2	The general problem of describing syntax, Formal methods of describing syntax, attribute grammars	2	6
3	Lexical analysis, the parsing problem, Recursive-Descent parsing	2	6
4	Variables, the concepts of binding, Type checking, Strong typing, Scope and lifetime, Referencing environments	1	3
5	Primitive Data Types, Different structures, Character string types, User-defined ordinal types, Arrays types, Record types, Union types, Pointer and reference types	1	3
6	Arithmetic expressions, Overloaded operators Types conversions,	1	3
7	Relational and Boolean expressions, short-circuit evaluation, Assignment statements, Mixed-mode assignment	1	3
8	Design issues for subprograms, Local referencing environments,	1	3
9	Parameter-passing methods, Parameters that are subprogram names, Generic subprograms, Coroutines	1	3
10	Implementing simple subprograms, Implementing subprograms with stack-dynamic local variables, Nested subprograms	2	6
11	Blocks, Implementing dynamic scoping	1	3
	Total	**14**	**42**

Textbook:

- Sebesta, Robert W. Concepts of programming languages. Boston: Pearson, 2016.

Essential References:

- Programming Language Concepts, Ghezzi and Jazayeri, Wse, 2008.
- Concepts in Programming Languages, Mitchell, Cambridge University Press, 2013.
- Programming Language Concepts and Paradigms, Watt, Watt, Frindlay, And Hughes, Prentice Hall, 1990.
- Programming Languages: Concepts and Constructs, Sethi, Addison-Wesley, 1989.
- Concepts of Programming Languages, Elson, Science Research Associates, 1973.
- Essentials of Programming Languages, Friedman, Wand, And Haynes, MIT Press, 2001.