Level 5 Diploma in Programming (601) 157 Credits

Unit: Programming Principles & Paradigms **Exam Paper No.:** 1	**Guided Learning Hours:** 280 **Number of Credits:** 28
Prerequisites: QBasic, Pascal or other programming language knowledge	**Corequisites:** A pass or higher in Diploma in System Design or equivalence.
Aim: The unit analyses programming languages and paradigms, the components that comprise them, and the principles of language design, all through the analysis and comparison of a variety of languages (e.g., Pascal, C++, PROLOG, ML). This unit is intended to broaden learners' experience beyond traditional imperative programming and provide a framework for understanding what makes a programming language useful. Learners learn about the basic components of programming languages and how they have evolved over time. This unit is for learners interested in high-level programming languages and their formal semantics. Such study enables precise reasoning about programs, their efficient implementation and easy reuse, as will be discussed in the unit. The materials to be covered include operational semantics, denotational semantics, and axiomatic semantics. The unit will consider imperative programming languages, functional programming languages, object-oriented programming languages, logic programming languages, and higher-level languages with sets and maps. Topics covered include type systems, abstraction mechanisms, declarativeness, efficient implementations, concurrency and parallelism. The unit objectives are: to provide an introduction to formalisms for specifying syntax and semantics of programming languages, including an introduction to the theory of formal languages; to provide an exposure to core concepts and principles in contemporary programming languages, and to explore various important programming methodologies, such as functional programming, logic programming, programming with abstract data types, and object-oriented programming; to provide a foundation in the concepts and implementation of modern programming languages by implementing imperative, functional, logic, and object-oriented programming paradigms. Upon completion of this unit, learners should be able to: understand language definition, abstractions, paradigms, basic knowledge about the major design principles; understand functional programming paradigm; understand the functional programming model – Lambda Calculus; be able to write functional programs in Lisp; understand logic programming paradigm principles – resolution and unification. be able to write Prolog programs; understand object-oriented programming principles – inheritance and dynamic binding; understand syntax definition and parsing techniques, including regular expressions, context-free grammars, parse trees and abstract syntax trees, syntax diagrams, and recursive descent parsing techniques; understand semantic definition, including attributes, binding and scoping.	
Required Materials: Recommended learning resources.	**Supplementary Materials:** Lecture notes and tutor extra reading recommendations.
Special Requirements: Topics are complicated; mostly written in abstract form, hence learners have to read a lot outside class time.	
Intended Learning Outcomes: 1. An overview of underlying principles as well as practical techniques used to design programs.	**Assessment Criteria:** 1.1 Analyse the properties of programming languages 1.2 Define programming paradigm 1.3 Describe event handling, concurrency and correctness 1.4 Explain the importance of programming 1.5 Outline programming design constraints 1.6 Distinguish compilers and interpreters 1.7 Describe the syntax of a programming language and why programming language syntax is important 1.8 Define programming language grammar 1.9 Describe extended BNF syntax 1.10 Describe programming languages syntax

	1.11 Analyse how compliers and interpreters work
2. The tools for illustrating the formal syntax and semantics of programming languages provide the basic mathematical techniques.	2.1 Describe *chomsky* hierarchy theorem 2.2 Explain lexical analysis and parsing tools 2.3 Define syntactic analysis 2.4 Analyse how programming languages are able to use and process named variables and their contents 2.5 Define reserved words 2.6 Outline characteristics of variables 2.7 Describe program scope 2.8 Analyse data structures for symbol tables 2.9 Describe programming reference environment 2.10 Describe dynamic scoping 2.11 Define name visibility, lifetime and overloading
3. How the value-level approach to programming invites the study of the space of values under the value-forming operations, and of the algebraic properties.	3.1 Describe type errors 3.2 Distinguish statically and dynamically typing 3.3 Distinguish basic and nonbasic types 3.4 Analyse subtypes, type equivalence and function types 3.5 Define polymorphism 3.6 Explain the syntax, type system and operational semantics and demonstrate the syntax; the semantics; the type system; the features 3.7 Demonstrate using expression semantics 3.8 Describe the program objects and values 3.9 Distinguish assignment statement vs assignment expression 3.10 Explain flow control semantics 3.11 Describe exception handling
4. Functions (procedures, subroutines, methods) facilities offered by a programming language.	4.1 Define a function 4.2 Describe function parameters 4.3 Describe parameter passing mechanisms 4.4 Demonstrate how to implement recursive functions 4.5 Analyse function declarations 4.6 Distinguish function call vs function return
5. Memory management and demonstrate the basic memory management issues that programmers face.	5.1 Identify and outline areas of memory 5.2 Describe arrays 5.3 Explain garbage collection 5.4 Outline an overview of the memory management techniques that are available 5.5 Compare and contrast functional programming with more traditional imperative (procedural) programming 5.6 Define imperative programming 5.7 Analyse procedural abstraction 5.8 Analyse imperative programming techniques

6. How developers use object-oriented programming techniques to implement frameworks such that the unique parts of an application can simply inherit from re-existing classes in the framework.	6.1 Analyse abstract data types 6.2 Identify characteristics of an object 6.3 Define and describe OOP languages 6.4 Describe how Application framework is best with graphical user interfaces (GUIs) 6.5 Describe the design, evolution and reuse of object-oriented application frameworks.
7. How functional programming emphasises the evaluation of expressions rather than the execution of commands.	7.1 Define functional programming 7.2 Compare and contrast functional and imperative programming 7.3 Describe lambda calculus 7.4 Describe Haskell encoding 7.5 Explain how the study of semantics also provides new ways of reasoning about the correctness of programs 7.6 Discuss the concept of Prolog programming language 7.7 Analyse the theory and practice of logic programming
8. How in event-driven programming responds to events paradigm in which the flow of the program is determined by events such as user actions.	8.1 Describe event-based programming paradigm 8.2 Distinguish event-driven, imperative and functional programming 8.3 Describe Graphical User Interface (GUI) applications 8.4 Explain how event-driven programming works
9. How in a concurrent program, several streams of operations may execute concurrently.	9.1 Outline the concurrency concepts 9.2 Describe synchronisation strategies 9.3 Describe synchronisation in Java
Methods of Evaluation: A 2½-hour written examination paper with five essay questions, each carrying 20 marks. Candidates are required to answer all questions. Candidates also undertake project/coursework in Programming Principles & Paradigms with a weighting of 100%.	

Recommended Learning Resources: Programming Principles & Paradigms

Text Books	• Programming Languages: Principles and Paradigms by Allen B Tucker and Robert Noonan ISBN-10: 0071254390 • Programming Language: Principles and Paradigms by Adesh Pandey ISBN-10: 1842653911 • Programming Languages: Principles and Paradigms by Maurizio Gabbrielli and Simone Martini ISBN-10: 1848829132
Study Manuals 	BCE produced study packs
CD ROM 	Power-point slides
Software 	Lite Programming Language