# PRINCIPLES OF PROGRAMMING LANGUAGES (CS515PE)

## COURSE PLANNER

### I.COURSE OVERVIEW:

Study of programming languages requires an examination of formal methods of describing the syntax and semantics of programming languages. Also ,implementation techniques for various language constructs such as lexical and syntax analysis, implementation of subprogram linkage and implementation of various programming languages are to be discussed. To briefly describe various programming paradigms. To provide conceptual understanding of High level language design and implementation. To introduce the power of scripting languages.

### II.PRE-REQUISITES:

This course requires basic computer knowledge and programming languages like C.

### III. COURSE OBJECTIVIES:

The following are the list of potential benefits of studying principles of programming language course.

1. To introduce the various programming paradigms.
2. To understand the evolution of programming languages.
3. To understand the concepts of OO languages, functional languages, logical and scripting languages.
4. To introduce the principles and techniques involved in design and implementation of modern programming languages.
5. To introduce the notations to describe the syntax and semantics of programming languages.
6. To introduce the concepts of concurrency control and exception handling.
7. To introduce the concepts of ADT and OOP for software development.

### IV. COURSE OUTCOMES:

| | Course Outcomes (CO) | Knowledge Level (Blooms Level) |
|---|---|---|
| **CO1** | ***Understand*** to express syntax and semantics in formal notation. | L2: Understand |
| **CO2** | ***Employ*** to apply suitable programming paradigm for the application. | L3: Apply |
| **CO3** | ***Design*** to program in different language paradigms and evaluate their relative benefits | L6: Create |
| **C04** | **Understand** the programming paradigms of modern programming languages. | L2: Understand |
| **C05** | **Understand** the concepts of ADT and OOP. | L2: Understand |
| **C06** | ***Knowledge*** to compare the features of various programming languages. | L1: Remember |

### V. HOW PROGRAMS OUTCOMES ARE ASSESSED:

| Program Outcomes (POs) | | Level | Proficiency assessed by |
|---|---|---|---|
| PO1 | **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. | **3** | Assignments |

| Program Outcomes (POs) | | Level | Proficiency assessed by |
|---|---|---|---|
| PO2 | **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences. | **3** | Assignments |
| PO3 | **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations. | **2** | Open ended experiments / |
| PO4 | **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. | **2** | Open ended experiments / |
| PO5 | **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations. | **1** | Mini Project |
| PO6 | **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice. | - | -- |
| PO7 | **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development. | - | -- |
| PO8 | **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. | - | -- |
| PO9 | **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings. | - | -- |
| PO10 | **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions. | **1** | Seminars / Term Paper |
| PO11 | **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments. | - | -- |

| Program Outcomes (POs) | | Level | Proficiency assessed by |
|---|---|---|---|
| PO12 | **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change. | **2** | Competitive Examinations |

**VI. HOW PROGRAM SPECIFIC OUTCOMES ARE ASSESSED:**

| Program Specific Outcomes (PSOs) | | Level | Proficiency assessed by |
|---|---|---|---|
| PSO1 | **Software Development and Research Ability:** Ability to understand the structure and development methodologies of software systems. Possess professional skills and knowledge of software design process. Familiarity and practical competence with a broad range of programming language and open source platforms. Use knowledge in various domains to identify research gaps and hence to provide solution to new ideas and innovations. | **3** | Lectures, Assignments |
| PSO2 | **Foundation of mathematical concepts:** Ability to apply the acquired knowledge of basic skills, principles of computing, mathematical foundations, algorithmic principles, modeling and design of computer- based systems in solving real world engineering Problems. | **2** | Mini Projects / Experiments |
| PSO3 | **Successful Career:** Ability to update knowledge continuously in the tools like Rational Rose, MATLAB, Argo UML, R Language and technologies like Storage, Computing, Communication to meet the industry requirements in creating innovative career paths for immediate employment and for higher studies. | **2** | Experiments / Tools |

**VII.SYLLABUS:**

**UNIT- I :**

**Preliminary Concepts**: Reasons for studying concepts of programming languages, programming domains, language evaluation criteria, influences on language design, language categories, language design trade-offs, implementation methods, programming environments, Evolution of Major Programming Languages.

**Syntax and Semantics**: General problem of describing syntax, formal methods of describing syntax, attribute grammars, describing the meanings of programs

**UNIT- II:**

Names, Bindings, and Scopes**:** Introduction, names, variables, concept of binding, scope, scope and lifetime, referencing environments, named constants

Data types: Introduction, primitive, character, string types, user defined ordinal types, array, associative arrays, record, tuple types, list types, union types, pointer and reference types, type checking, strong typing, type equivalence

Expressions and Statements**:** Arithmetic expressions, overloaded operators, type conversions, relational and boolean expressions, short- circuit evaluation, assignment statements, mixed-mode assignment

Control Structures **–** introduction, selection statements, iterative statements, unconditional branching, guarded commands**.**

**UNIT- III:**

Subprograms: Fundamentals of subprograms, design issues for subprograms, local referencing environments, parameter passing methods, parameters that are subprograms, calling subprograms indirectly, overloaded subprograms, generic subprograms, design issues for functions, user defined overloaded operators, closures, co routines

Implementing subprograms: General semantics of calls and returns, implementing simple subprograms, implementing subprograms with stack-dynamic local variables, nested subprograms, blocks, implementing dynamic scoping

Abstract Data types**:** The concept of abstraction, introductions to data abstraction, design issues, language examples, parameterized ADT, encapsulation constructs, naming encapsulations

**UNIT- IV:**

Object Oriented Programming**:** Design issues for OOP, OOP in Smalltalk, C++, Java, Ada 95, Ruby, Implementation of Object-Oriented constructs.

Concurrency**:** Introduction, introduction to subprogram level concurrency, semaphores, monitors, message passing, Ada support for concurrency, Java threads, concurrency in functional languages, statement level concurrency. Exception Handling and Event Handling: Introduction, exception handling in Ada, C++, Java, introduction to event handling, event handling with Java and C#.

**UNIT-V :**

**Functional Programming Languages**: Introduction, mathematical functions, fundamentals of functional programming language, LISP, support for functional programming in primarily imperative languages, comparison of functional and imperative languages

**Logic Programming Language**: Introduction, an overview of logic programming, basic elements of prolog, deficiencies of prolog, applications of logic programming.

**Scripting Language**: Pragmatics, Key Concepts, Case Study : Python – Values and Types, Variables , Storage and Control, Bindings and Scope, Procedural Abstraction, Data Abstraction, Separate Compilation, Module Library. (Text Book 2)

**TEXT BOOKS:**

1. Concepts of Programming Languages, Robert .W. Sebesta 10th edition, Pearson Education.
2. Programming Language Design Concepts, D. A. Watt, Wiley India Edition.

**REFERENCE BOOKS:**

1. Programming Languages, A.B. Tucker, R.E. Noonan, TMH.
2. Programming Languages, K. C. Louden and K A Lambert., 3rd edition, Cengage Learning.
3. Programming Language Concepts, C Ghezzi and M Jazayeri, Wiley India.
4. Programming Languages 2nd Edition Ravi Sethi Pearson.
5. Introduction to Programming Languages Arvind Kumar Bansal CRC Press.

## VII.LESSON PLAN:

| S. NO. | WEEK | Unit | Topics | Topics to be covered | Link for PPT | Link for PDF | Link for Small Projects/ Numericals(if any) | Course Learning Outcomes | Teaching Methodology | References |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | Reasons for studying concepts of programming language,programming domani | • Introduction,<br>• Evolution Needs<br>• Different types of languages in brief<br>• Increased capacity to express ideas<br>• Scientific Applications<br>• Business Applications<br>• Artificial Intelligence<br>• Systems Programming | https://docs.google.com/presentation/d/1UuTTL1np7RQfPEEkg62OZohRGvWDP7O0/edit?usp=sharing&ouid=107625280176593675505&rtpof=true&sd=true | https://drive.google.com/file/d/1SbKraAoKsvLKwr-WaDdCn2geLMxe4HED/view?usp=sharing | **NA** | Understand | Chalk and talk | TI |
| 2 | | 1 | language evaluation criteria | • Readability<br>• Overall Simplicity<br>• Orthogonality<br>• Data Types<br>• Syntax Design<br>• Write ability | | | **NA** | Understand | Chalk and talk | TI |
| 3 | | 1 | Influences on Language Design, Language Categories | • Computer Architecture<br>• Programming Design Methodologies<br>• Procedural<br>• Object oriented | | | **NA** | Understand | Chalk and talk | TI |
| 4 | 2 | 1 | Language Design Trade-Offs | • Reliability<br>• Cost of execution | | | **NA** | Understand | Chalk and talk | TI |
| 5 | | 1 | Implementation Methods, Programming Environments | • Internal memory<br>• Processor | | | **NA** | Understand | Chalk and talk | TI |
| 6 | | 1 | General Problem of Describing Syntax and Semantics, | • Language Recognizers<br>• Language Generators | | | **NA** | Understand | Chalk and talk | TI |
| 7 | 3 | 1 | Formal Methods of Describing Syntax, | • Backus-Naur Form and Context-Free Grammars<br>• Context-Free Grammars<br>• Origins of Backus-Naur Form<br>• Fundamentals<br>• Describing Lists<br>• Grammars and Derivations | | | **NA** | Understand | Chalk and talk | TI |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | • Parse Trees<br>• Ambiguity<br>• Extended BNF<br>• Grammars and Recognizers | | | | | | |
| 8 | 3 | 1 | Attribute Grammars, Describing the Meanings of Programs | • Static Semantics<br>• Basic Concepts<br>• Attribute Grammars Defined<br>• Intrinsic Attributes<br>• Examples of Attribute Grammars<br>• operational Semantics<br>• The Basic Process<br>• Denotational Semantics<br>• Two Simple Examples<br>• The State of a Program | | | **NA** | Understand | Chalk and talk | TI |
| 9 | | 2 | Introduction, Names, Variables, Concept of Binding | • Introduction<br>• Design Issues<br>• Binding of Attributes to Variables<br>• Type Bindings<br>• Storage Bindings and Lifetime | https://docs.google.com/presentation/d/1ipW02xkAtKVy2Gf8D0NseC-bR89imA-b/edit?usp=sharing&ouid=107625280176593675505&rtpof=true&sd=true | https://drive.google.com/file/d/1ErZ7Sv4gPzGiX3aYiHSmy0tk3IY2-k84/view?usp=sharing | **NA** | Understand | Chalk and talk | TI |
| 1 | 4 | | mock test-1 | | | | **NA** | Understand | Chalk and talk | TI |
| 1 | | 2 | Scope, Scope and Lifetime,Referencing Environments, Named Constants | • Static Scope<br>• Blocks<br>• Declaration Order<br>• Global Scope<br>• Dynamic Scope<br>• Different types of lifetimes• Examples<br>• Example with global variable | | | **NA** | Understand | Chalk and talk | TI |
| 1 | | 2 | Data Types: Introduction, Primitive Data Types, Character String Types, User Defined Ordinal Types, | • Primitive Data Types<br>• Boolean Types<br>• Character Types<br>• Character String Types<br>• Enumeration Types<br>• Evaluation<br>• Subrange Types<br>• Implementation of User-Defined Ordinal Types | | | **NA** | Understand | Chalk and talk | TI |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 2 | Tuple Types, List Types, Pointer and Reference Types, Type | • Definitions of Records<br>• References to Record Fields<br>• Evaluation<br>• Implementation of Record Types | | | **NA** | Understand | Chalk and talk | TI |
| | | | | • Discriminated Versus Free Unions | | | | | | |
| 1 | | 2 | Checking, Strong Typing, Type Equivalence | • Pointer Operations<br>• Pointer Problems<br>• Dangling Pointers<br>• Pointers in Ada<br>• Pointers in C and C++<br>• Implementation of Pointer and Reference Types | | | **NA** | Understand | Chalk and talk | TI |
| 1 | | 2 | Short Circuit Evaluation, Assignment Statements | • Simple Assignments<br>• Conditional Targets<br>• Compound Assignment Operators<br>• Unary Assignment Operators<br>• Assignment as an Expression<br>• Multiple Assignments<br>• Assignment in Functional Programming Languages | | | **NA** | Understand | Chalk and talk | TI |
| 1 | 6 | 2 | Mixed-Mode Assignment,Control Structures – Introduction,Selection Statements | Simple Assignments<br>Conditional Targets<br>Compound Assignment Operators<br>Unary Assignment Operators | | | **NA** | Understand | Chalk and talk | TI |
| 1 | | 2 | Iterative Statements | Simple Assignments<br>Conditional Targets<br>Compound Assignment Operators<br>Unary Assignment Operators | | | **NA** | Understand | Chalk and talk | TI |
| 1 | | 2 | Unconditional,Branching, Guarded Commands. | Assignment as an Expression<br>Multiple Assignments<br>Assignment in Functional Programming Languages | | | **NA** | Understand | Chalk and talk | TI |
| **MID-I EXAMINATION ( From 08-11-2021 )** | | | | | | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 7 | 3 | Fundamentals of Sub-Programs, Design Issues for Subprograms, | • General Subprogram Characteristics<br>• Basic Definitions<br>• Parameters<br>• Procedures and Functions | https://docs.google.com/presentation/d/17LLKPruNBP-KuzViJlHQPXX-MzTgcXil/edit?usp=sharing&ouid=107625280176593675505&rtpof=true&sd=true | https://drive.google.com/file/d/1JA6waK_CPoBxN1OJC2P-roReKuRDCdKn/view?usp=sharing | | | | |
| 2 | | 3 | Local Referencing Environments, Parameter Passing Methods, | • Local Variables<br>• Nested Subprograms<br>• Semantics Models of Parameter Passing<br>• Implementation Models of Parameter Passing<br>• Pass-by-Value<br>• Pass-by-Result<br>• Pass-by-Reference<br>• Pass-by-Name | | | **NA** | Understand | Chalk and talk | TI |
| 2 | 8 | 3 | Parameters that Are Subprograms, Calling Subprograms Indirectly, Overloaded Subprograms, Generic Subprograms | • Generic Functions in C++<br>• Generic Methods in Java 5.0<br>• Generic Methods in C# 2005<br>• Generic Functions in F# | | | **NA** | Understand | Chalk and talk | TI |
| 2 | | 3 | Design Issues for Functions, User Defined Overloaded Operators, | • Functional Side Effects<br>• Types of Returned Values<br>• Number of Returned Values<br>• User-Defined Overloaded Operators | | | **NA** | Understand | Chalk and talk | TI |
| 2 | | 3 | Closures, Coroutines | Number of Returned Values<br>User-Defined Overloaded Operators | | | **NA** | Understand | Chalk and talk | TI |
| 2 | | 3 | General Semantics of Calls and Returns, Implementing Simple Subprograms | Implementing “Simple” Subprograms<br>An Example Without Recursion<br>Recursion<br>Static Chains<br>The basics | | | **NA** | Understand | Chalk and talk | TI |
| 2 | 9 | | mock test-2 | | | | | | | |
| 2 | | 3 | Implementing Subprograms with Stack-Dynamic Local Variables, Nested Subprograms | • Implementing “Simple” Subprograms<br>• An Example Without Recursion<br>• Recursion<br>• Static Chains | | | **NA** | Understand | Chalk and talk | TI |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Blocks,Implementing Dynamic Scoping | • The basics | | | | | | |
| 2 | | 3 | The Concept of Abstraction, Introductions to Data Abstraction, Design Issues, | • Introduction to Data Abstraction<br>• Floating-Point as an Abstract Data Type<br>• User-Defined Abstract Data Types | | | **NA** | Understand | Chalk and talk | TI |
| 2 | 10 | 3 | Language Examples, Parameterized ADT, Encapsulation Constructs, Naming Encapsulations | • Abstract Data Types in Ada<br>• Abstract Data Types in C++<br>• Abstract Data Types in Obj.-C<br>• Abstract Data Types in Java<br>• Abstract Data Types in C#<br>• Abstract Data Types in Ruby | | | **NA** | Understand | Chalk and talk | TI |
| 2 | | 4 | Concurrency: Introduction, Introduction to Subprogram Level Concurrency, Semaphores, Monitors | • Multiprocessor Architectures<br>• Categories of Concurrency<br>• Motivations for the Use of Concurrency<br>• Introduction to Subprogram-Level Concurrency<br>• Design Issues | https://docs.google.com/presentation/d/17LLKPruNBP-KuzViJlHQPXX-MzTgcXil/edit?usp=sharing&ouid=107625280176593675505&rtpof=true&sd=true | https://drive.google.com/file/d/12HAsJ_BN3N3kAur553f6E2ii8FJOhWzB/view?usp=sharing | | | | |
| 3 | | 4 | Message Passing, Java Threads, Concurrency in Function Languages, Statement Level Concurrency | • Introduction<br>• Competition Synchronization<br>• Cooperation Synchronization<br>• The Concept of Synchronous Message Passing | | | **NA** | Understand | Chalk and talk | TI |
| 3 | 11 | 4 | Exception Handling and Event Handling: Introduction, Exception Handling in Ada, C++, Java,Introduction to Event Handling, Event Handling with Java and C#. | • Basic Concepts<br>• Design Issues<br>• Exception Handling in Ada<br>• Binding Exceptions to Handlers<br>• Other Design Choices | | | **NA** | Understand | Chalk and talk | TI |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 3 | | 5 | Functional Programming Languages: Introduction, Mathematical Functions, Fundamentals of Functional Programming Language | • Introduction<br>• Mathematical Functions<br>• Simple Functions<br>• Functional Forms<br>• The First Functional Programming Language: LISP<br>• Data Types and Structures | | | **NA** | Understand | Chalk and talk | TI |
| 3 | | 5 | LISP, Support for Functional Programming in Primarily Imperative,Languages, Comparison of Functional and Imperative Languages | • The First LISP Interpreter<br>• Origins of Scheme<br>• The Scheme Interpreter<br>• Primitive Numeric Functions<br>• Defining Functions<br>• Output Functions | https://docs.google.com/presentation/d/1X8EUM-1hhsaSXbPfunfJcdaH0KZILLD1/edit?usp=sharing&ouid=107625280176593675505&rtpof=true&sd=true | **https://drive.google.com/file/d/12HAsJ_BN3N3kAur553f6E2ii8FJOhWzB/view?usp=sharing** | **NA** | Understand | Chalk and talk | TI |
| 3 | 12 | 5 | Logic Programming Language: Introduction, an Overview of Logic Programming, Basic Elements of Prolog | • A Brief Introduction to Predicate Calculus<br>• Propositions<br>• Clausal Form<br>• Predicate Calculus and Proving Theorems | | | **NA** | Understand | Chalk and talk | TI |
| 3 | | 5 | Applications of Logic Programming. | The Basic Elements of Prolog<br>Terms<br>Fact Statements<br>Rule Statements<br>Goal Statements<br>The Inferencing Process of Prolog | | | **NA** | Understand | Chalk and talk | TI |
| 3 | | 5 | Scripting Language: Pragmatics, Key Concepts, Case Study: Python – Values and Types | • Relational Database Management Systems<br>• Expert Systems | | | **NA** | Understand | Chalk and talk | TI |
| 3 | 13 | 5 | Variables,Storage and Control | • Resolution Order Control | | | **NA** | Understand | Chalk and talk | TI |
| 3 | | 5 | Bindings and Scope, Procedural Abstraction, | • The First LISP Interpreter<br>• Origins of Scheme<br>• The Scheme Interpreter<br>• Primitive Numeric Functions<br>• Defining Functions<br>• Output Functions | | | **NA** | Understand | Chalk and talk | TI |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 3 | | 5 | Data Abstraction, Separate Compilation | • A Brief Introduction to Predicate Calculus<br>• Propositions<br>• Clausal Form<br>• Predicate Calculus and Proving Theorems | | | **NA** | Understand | Chalk and talk | TI |
| 4 | 14 | 5 | Module Library | | | | **NA** | Understand | Chalk and talk | TI |
| 4 | | | presentations | | | | **NA** | Understand | Chalk and talk | TI |
| **MID-II EXAMINATION ( From 10-01-2022 )** | | | | | | | | | | |

## IX. MAPPING COURSE OUTCOMES LEADING TO THE ACHIEVEMENT OF PROGRAM OUTCOMES AND PROGRAM SPECIFIC OUTCOMES

| Course Outcomes | Program Outcomes (PO) | | | | | | | | | | | | Program Specific Outcomes (PSO) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** | **PSO1** | **PSO2** | **PSO3** |
| **CO1** | 2 | 2 | 1 | - | - | - | - | - | - | 1 | - | 1 | 2 | 2 | 2 |
| **CO2** | 3 | 3 | 2 | 2 | 1 | - | - | - | - | 1 | - | 2 | 3 | 2 | 2 |
| **CO3** | 3 | 3 | 3 | 1 | 1 | - | - | - | - | 1 | - | 2 | 3 | 1 | 1 |
| **CO4** | 3 | 3 | 2 | 2 | 1 | - | - | - | - | 1 | - | 2 | 3 | 2 | 2 |
| **CO5** | 3 | 3 | 2 | 2 | 1 | - | - | - | - | 1 | - | 2 | 3 | 2 | 2 |
| **CO6** | 2 | 2 | - | - | - | - | - | - | - | 1 | - | 1 | 2 | 1 | 1 |
| **AVG** | **2** | **2** | **2** | **1.75** | **1.0** | **-** | **-** | **-** | **-** | **1.0** | **-** | **1.67** | **2.67** | **1.67** | **1.67** |

## X.QUESTION BANK:

### UNIT- 1

### Short Answer Questions

| QUESTIONS | Blooms taxonomy level | Course Outcome |
|---|---|---|
| 1. Define imperative programming language? | **Understand** | **CO1** |
| 2. Differentiate between special purpose and general purpose languages? | **Understand** | **C01** |
| 3. Differentiate between Syntax and Semantics? | **Knowledge** | **CO1** |
| 4. Write BNF and EBNF grammar for expressions? | **Knowledge** | **CO1** |

### Long Answer Questions

| QUESTIONS | Blooms taxonomy level | Course Outcomes |
|---|---|---|
| 1. Explain list of criteria for language evaluation**?** | **Create** | **CO1** |

| | | |
|---|---|---|
| 2. Explain the reasons for studying concepts of programming language? | **Analyze** | **CO1** |
| 3. Explain the concept of orthogonality in program language Design? | **Understanding** | **CO1** |
| 4. Explain in detail about attribute grammar for simple assignment statement | **Create** | **CO1** |
| 5. Describe three advantages of LR parser? | **Analyze** | **CO1** |

**UNIT- 2**

**Short Answer Questions**

| QUESTIONS | Blooms taxonomy level | Course Outcomes |
|---|---|---|
| 1. Define data type and importance of data types? | **Understand** | **CO2** |
| 2. Explain use of C++ reference type? | **Understand** | **CO2** |
| 3. What are the advantages and disadvantages of implicit declaration? | **Knowledge** | **CO2** |
| 4. What are the advantages and disadvantages of implicit declaration? | **Knowledge** | **CO2** |
| 5. Explain associative arrays, their structure and operations? | **Analyze** | **CO2** |

**Long Answer Questions**

| QUESTIONS | Blooms taxonomy level | Course Outcomes |
|---|---|---|
| 1. Discuss about various data types? | **Create** | **CO2** |
| 2. Define an array. Explain the design issues and different types of arrays? | **Analyze** | **CO2** |
| 3. Explain unconditional statements supported by different programming languages? | **Understanding** | **CO2** |
| 4. Describe the various control statements in programming languages. | **Create** | **CO2** |
| 5. Discuss guarded commands in detail? | **Analyze** | **CO2** |

**UNIT -3**

**Short Answer Questions**

| QUESTIONS | Blooms taxonomy level | Course Outcomes |
|---|---|---|
| 1. Define how C language deals with scope and lifetime of a variable? | **Understand** | **CO3** |
| 2. Define subprogram and explain its general characteristics? | **Understand** | **CO3** |

| | | |
|---|---|---|
| 3. Explain about parameter passing methods? | **Knowledge** | **CO3** |
| 4. Write about co routines? | **Knowledge** | **CO3** |
| 5. Write about overloaded Programs | **Knowledge** | **CO4** |
| 1. Explain associative arrays, their structure and operations? | **Analyze** | **CO4** |

**Long Answer Questions**

| QUESTIONS | Blooms taxonomy level | Course Outcomes |
|---|---|---|
| 1. Define subprogram and explain the distinct categories of sub programs. | **Create** | **CO3** |
| 2. What is Generic Subprogram give few examples in Ada,C++,Java? | **Analyze** | **CO3** |
| 3. Write short note on overloaded sub programs. | **Understanding** | **CO3** |
| 4. Discuss how generic functions are implemented in C ++. | **Create** | **CO4** |
| 5. Explain how a sub program name can be passed as parameter to other sub programs. | **Analyze** | **CO4** |

**UNIT -4**

**Short Answer Questions**

| QUESTIONS | Blooms taxonomy level | Course Outcome |
|---|---|---|
| 1. What problems can occur using C to define abstract data types | **Understand** | **CO4** |
| 2. Distinguish between C++ class and ADA package | **Understand** | **CO4** |
| **3.** Distinguish C++ throw specification and throw clause in Java | **Knowledge** | **CO4** |
| 4. Explain the uses of exception handling in programming languages? | **Knowledge** | **CO5** |
| 5. Write the applications of logic programming? | **Analyze** | **CO5** |

**Long Answer Questions**

| QUESTIONS | Blooms taxonomy level | Course outcome |
|---|---|---|
| 1. Explain the object oriented programming in small talk, C++ and Java? | **Create** | **CO4** |
| 2. Define semaphores. Explain how cooperation and competition synchronization are implemented using semaphores? | **Analyze** | **CO4** |
| 3. Explain the applications of Logic programming? | **Understanding** | **CO4** |
| 4. Define a task . Explain the different states of Task? | **Create** | **CO5** |
| 5. Explain dynamic binding in C++ and Java? | **Analyze** | **CO5** |

**UNIT -5**

**Short Answer Questions**

| QUESTIONS | Blooms taxonomy level | Course Outcome |
|---|---|---|
| 1. What is S expression and how it is evaluated with an example. | **Understand** | **CO5** |
| 2. Write a LISP function that calculates sum of numbers using a vector | **Understand** | **CO5** |
| **3.** List few characteristics of Python language? | **Knowledge** | **CO5** |
| 4. List few examples of scripting languages? | **Knowledge** | **CO6** |
| 5. List the draw backs of using an imperative language compared to FP? | **Knowledge** | **CO6** |

**Long Answer Questions**

| QUESTIONS | Blooms taxonomy level | Course Outcome |
|---|---|---|
| 1. Describe the following for LISP a)Data types and structures b)LISP interpreter<br>Compare Functional Languages with Imperative languages | **Create** | **CO5** |
| 2. What are the three features of Haskell that make it significantly different from Scheme | **Analyze** | **CO5** |
| 3. Explain procedural and data abstraction in python | **Understanding** | **CO5** |
| 4. Discuss briefly about HTML parsing and CGI argument parsing. | **Create** | **CO6** |
| 5. What is scripting and explain the characteristics of scripting languages | **Analyze** | **CO6** |

**OBJECTIVE QUESTIONS**

**UNIT-1**

1. The following is the widely used programming language developed for Artificial Intelligence Application

A)LISP b)FORTRAN c)COBOL d).ALGOL 602.

2. The following language require Interpreter
   a) C++ b). C c). COBOL d). APL
3. In C and C++ the asterisk (*) denotes the following operation.
   a)Dereferencing b) negation c) referencing d) address

4. In FORTRAN90, Loop parameters are evaluated
   a)Thrice b) only once c) twice d) every time

5.The following Type compatibility is described in Semantics
   a)Structured b) Static c) Denotional d) Dynamic

**UNIT-2**

**1.**The following language has pointer concept

a) Java b) c++ c)DHTML d) HTML

2. The first high level programming to included pointer variables was
   a). Fortran b) PL/1 c) ALGOL 60 d) ADA

3. The following variables should not appear in recursive functions
a)Register b) static c) auto d) Extern
4. PDA means
a). Pop down Automata b). Push down automata
c). Push Dip Automata d). Push Down Automatic
5. The type of the following operator ?: is
a). unary b). not an operator c). ternary d). binary

**UNIT-3**

1. A describes the interface to and the actions of the subprogram []
1. subprogram 2. Interface 3.scope 4.blocks
2. The caller is _____during execution of the called subprogram []
A) suspended (B) terminated (C) blocked (D) none
3. The period of time between an allocation and its subsequent disposal is called
(A)Life time (B) scope (C) binding (D) all
4. In java, object parameters are passed using[ ]
(A) call-by-name (B) call-by-value (C) call-by-reference (D) call-by-result
5. Variables defined inside the subprogram is called [ ]
A) Global variables B) Local Variables C) Parameter D)None

**UNIT-4**

1. _____ are used along with the variables in Prolog

A) quantifiers B)qualifiers C)terms D) B&D

2. Finding value to variable in prolog is

A) Unification B) Simplification C) Matching D)exceptional propagation

3. Java clause provides a mechanism for guaranteeing that some code will be executed how the execution of a try compound terminates.

A) Finally B) throws C) try D) catch

4. The categories of exceptions in Java are

a) checked b) unchecked c)constant d)a&b

5. Re-consideration of path is

a) Backtracking b) BFS c) forward chaining d) backward chaining

6. Logic programming languages are used in
a) RDBMS b)expert systems c)natural language processing d)all

**UNIT-5**

1. A static-scoped functional language with syntax that is closer to Pascal than to LISP

A)ML B) HASKELL C) C D) FORTRAN

2. _____Uses lazy evaluation (evaluate no sub expression until the value is needed)
A)ML B)HASKELL C) LISP D) PROLOG

3. Pure LISP has only two kinds of data structures atoms and [ ]
A). Arrays B). Lists C). variables D). Stack

4. The design of the functional languages is based on
1.mathematical functions 2. Predicate calculus 3. Relations 4.none

5. The first functional programming language
A). ML B). HASKELL C). LISP D). FORTRAN

**Fill in the blanks**

**UNIT-1**

1. C was developed by _____
2. BNF is _____
3. _____is the language of axiomatic semantics.
4. _____semantics was defined in conjunction with the development of a method to prove the correctness of programs.
5. _____can be used to accept the sentence of a language.

**UNIT-2**

1._____ refers to a data type in which all the values compromises of a sequence of character
2._____keyword is used to define global variables visible in all the object modules
3._____is a compound expression that contains three expressions.
4.In C switch –case statement the default expression type c a n be _____.
5._____language support independent compilation

**UNIT-3**

1._____ is the first part of the definition, including the name, the kind of subprogram, and the formal parameters
2.The _____is a subprogram's parameter profile and, if it is a function, its return type
3.C++ A special pointer type called reference type for _____
4.Java :All parameters are _____
5.C _____is achieved by using pointers as parameters

**UNIT-4**

1. Nearly all programming languages support process abstraction with _____
2. _____does not currently support parameterized classes
3. A class that inherits is a ____or _____
4. The entire collection of methods of an object is called its _____.
5. Task execution control is maintained by a program called the_____

**UNIT- 5**

1._____is strongly typed (whereas Scheme is essentially type less) and has no type coercions
2._____ is used for throw-away programs
3._____ is a process of writing small sized programs so as to glue different software tools
4._____ is a open source language
5._____is the only parameter passing method by Python.

**GATE: Not Applicable**

**IES: Not Applicable**

**XI.WEBSITES:**

1.www.nptel.iitm.ac.in/video.php?subjectId=106102067
2. www.cs.cmu.edu/~rwh/courses/ppl/
3. http://www.apl.jhu.edu/~hall/lisp.html

4. http://www.swi-prolog.org/pldoc/refman/

**XII.EXPERT DETAILS:**

1. Dr. K. Gopinath
   Professor
   Computer Science & Automation (CSA),
   Indian Institute of Science (IISc), Bangalore 560012 INDIA
2. Dr. A. GOVARDHAN
   Professor in CSE &
   JNTU Hyderabad.
2. Dr. T. Srinivasulu Reddy, Professor in , JNTU Hyderabad.

**XIII. JOURNALS:**

**(National & International)**

1. SCP - Science of Computer Programming
2. TOPLAS - ACM Transactions on Programming Languages and Systems
3. JFP - Journal of Functional Programming
4. JLP - The Journal of Logic and Algebraic Programming
5. TPLP - Theory and Practice of Logic Programming
6. CL - Computer Languages, Systems & Structures
7. IJPP - International Journal of Parallel Programming
8. JOOP - Journal of Object-oriented Programming

**XIV.LIST OF TOPICS FOR STUDENT SEMINARS:**

1. Reasons for studying programming language
2. General problem of describing syntax and Axiomatic semantics for common programming language features
3. Pointer Reference types and applications in various programming languages.
4. Short circuit evaluation and Mixed mode assignment

**XV.CASE STUDIES/SMALL PROJECTS:**

Write a BNF grammar for e-mail addresses that can express the following examples:

morgan@cs.williams.edu
steele@java.com
Morgan.McGuire@williams.edu -
dingle@_.com 377..5@hotmail.com
president@whitehouse.gov
_underscorer_@slashdot.org
scott_mccann@2mail.f4st.111.org
and rejects the following:

bad#email.com hello@world
funny/symbol@none.gov jon@edu
illegal@domain.name whole@lota@at.com
empty@..com

Assume that the only legal top-level domains (TLDs) in this grammar are gov, edu, com, and org, and that there must be at least two period-separated names to the right of the @, and that those names must contain at least one character each. Assume that the only legal symbols in an e-mail address to the left and right of the @ are period,
underscore, and dash (minus). (This is all a simplification of real e-mail grammars to make the

problem easier. If you happen to know the real rules...forget them while you're working on this problem! You can find the real grammar in RFCs

1034 and 822)Remember to put quotes around terminals and angle-brackets around non-terminals. You may use the regular expression operators [ ] + * for convenience in addition to pure BNF grouping parentheses and the exclusive

or operator, |

. The following productions are provided:

<digit> ::= `0' | `1' | `2' | `3' | `4' | `5' | `6' | `7' | `8' | `9'

<alpha> ::= 'a' | `b' | `c' | `d' | `e' | `f' | `g' | `h' | `i' | `j' | `k' | `l' | `m' |

`n' | `o' | `p' | `q' | `r' | `s' | `t' | `u' | `v' | `w' | `x' | `y' | `z' |

'A' | `B' | `C' | `D' | `E' | `F' | `G' | `H' | `I' | `J' | `K' | `L' | `M' | `N'

| `O' | `P' | `Q' | `R' | `S' | `T' | `U' | `V' | `W' | `X' | `Y' | `Z'