CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Approved by AICTE - New Delhi. Affiliated to JNTUH and Accredited by NAAC & NBA)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE INSTRUCTOR NAME: B.Mamatha **ACADEMIC.YEAR:2023-24**

SUBJECT NAME: Principles of Programming Languages **CLASS ROOM NO:B-217&219**

E-MAIL ID: b.mamatha@cmrec.ac.in

CONTACT NO: 9550715635

SEM START AND SEM END DATES: 21-08-2023 TO 24-12-2023

CONTENTS OF COURSE FILE:

1. DEPARTMENT VISION & MISSION

Vision:

To produce globally competent and industry-ready graduates in Computer Science & Engineering by imparting quality education with the know-how of cutting-edge technology and holistic personality.

Mission:

1. To offer high-quality education in Computer Science & Engineering in order to build core competence for the graduates by laying a solid foundation in Applied Mathematics and program frameworkwith a focus on concept building.

2. The department promotes excellence in teaching, research, and collaborative activities to prepare graduates for a professional career or higher studies.

3. Creating an intellectual environment for developing logical skills and problem-solving strategies, thus developing, an able and proficient computer engineer to compete in the current global scenario.

2. LIST OF PEOs, POs AND PSOs

2.1 Program Educational Objectives (PEO):

PEO 1: Excel in professional career and higher education by acquiring knowledge of mathematical computing and engineering principles.

PEO 2: To provide an intellectual environment for analyzing and designing computing systems for technical needs.

PEO 3: Exhibit professionalism to adapt current trends using lifelong learning with legal and ethical responsibilities.

PEO 4: To produce responsible graduates with effective communication skills and multidisciplinary practices to serve society and preserve the environment.

2.2. Program Outcomes (POs):

Engineering Graduates will be able to satisfy these NBA graduate attributes:

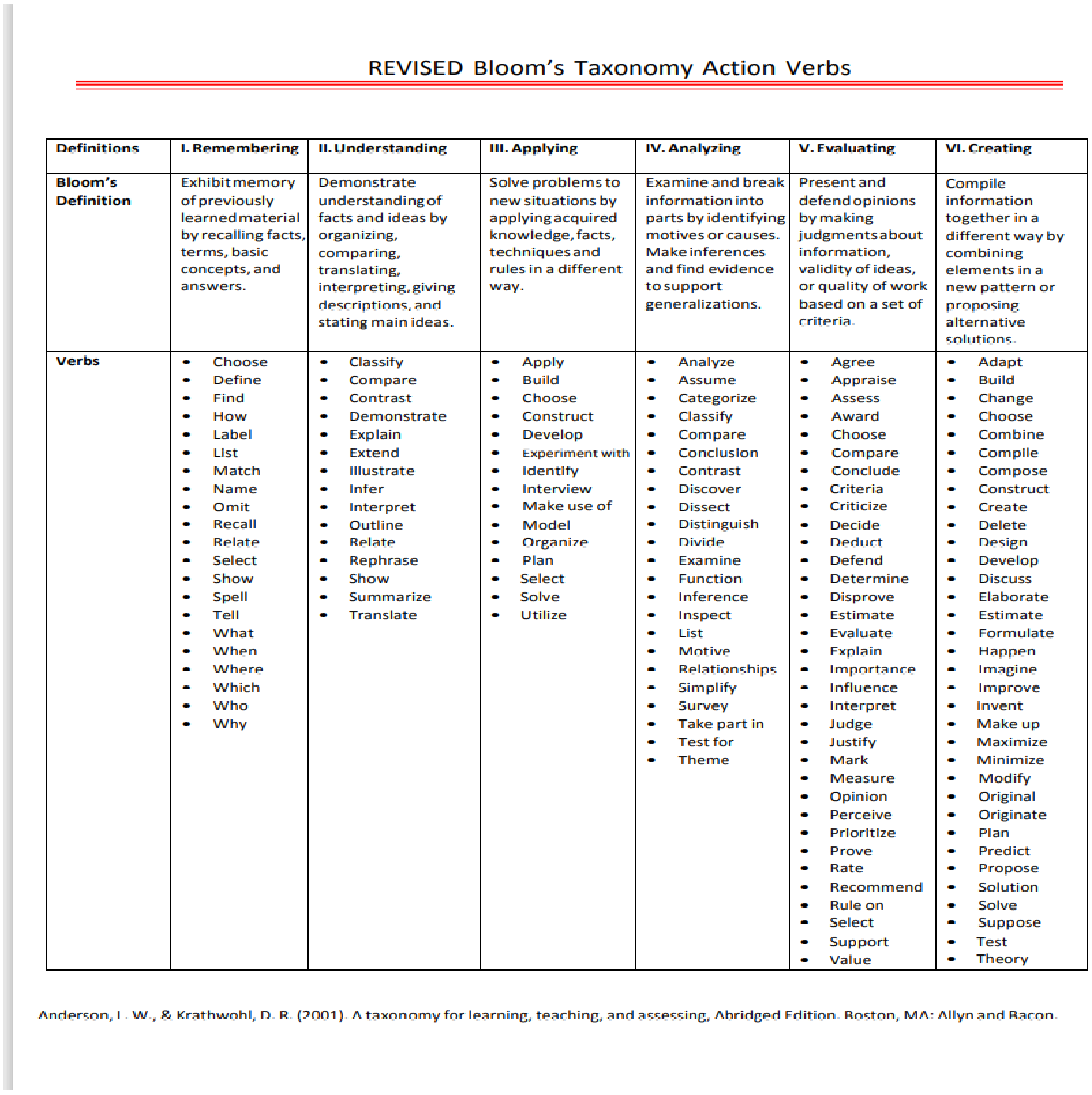
1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **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
4. **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
5. **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
6. **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
7. **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
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings
10. **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
11. **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
12. **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

2.3 Program Specific Outcomes (PSOs):

PSO1: Professional Skills and Foundations of Software development: Ability to analyze, design and develop applications by adopting the dynamic nature of Software developments.
PSO2: Applications of Computing and Research Ability: Ability to use knowledge in cutting edge technologies in identifying research gaps and to render solutions with innovative ideas..

3. LIST OF COS (ACTION VERBS AS PER BLOOM'S TAXONOMY)

CO1	**Explain** the important features of the Programming Languages.
CO2	**Compare** different Programming Domains.
CO3	**Evaluate** Merits and Demerits of a Particular Programming Language.
CO4	**Choose** Specific Programming Language for the **Development** of Specific Applications.
CO5	**Understand** and **Analyze** the Importance of Implementation Process.



REVISED Bloom's Taxonomy Action Verbs

Definitions	**I. Remembering**	**II. Understanding**	**III. Applying**	**IV. Analyzing**	**V. Evaluating**	**VI. Creating**
Bloom's Definition	Exhibit memory of previously learned material by recalling facts, terms, basic concepts, and answers.	Demonstrate understanding of facts and ideas by organizing, comparing, translating, interpreting, giving descriptions, and stating main ideas.	Solve problems to new situations by applying acquired knowledge, facts, techniques and rules in a different way.	Examine and break information into parts by identifying motives or causes. Make inferences and find evidence to support generalizations.	Present and defend opinions by making judgments about information, validity of ideas, or quality of work based on a set of criteria.	Compile information together in a different way by combining elements in a new pattern or proposing alternative solutions.
Verbs	• Choose	• Classify	• Apply	• Analyze	• Agree	• Adapt
	• Define	• Compare	• Build	• Assume	• Appraise	• Build
	• Find	• Contrast	• Choose	• Categorize	• Assess	• Change
	• How	• Demonstrate	• Construct	• Classify	• Award	• Choose
	• Label	• Explain	• Develop	• Compare	• Choose	• Combine
	• List	• Extend	• Experiment with	• Conclusion	• Compare	• Compile
	• Match	• Illustrate	• Identify	• Contrast	• Conclude	• Compose
	• Name	• Infer	• Interview	• Discover	• Criteria	• Construct
	• Omit	• Interpret	• Make use of	• Dissect	• Criticize	• Create
	• Recall	• Outline	• Model	• Distinguish	• Decide	• Delete
	• Relate	• Relate	• Organize	• Divide	• Deduct	• Design
	• Select	• Rephrase	• Plan	• Examine	• Defend	• Develop
	• Show	• Show	• Select	• Function	• Determine	• Discuss
	• Spell	• Summarize	• Solve	• Inference	• Disprove	• Elaborate
	• Tell	• Translate	• Utilize	• Inspect	• Estimate	• Estimate
	• What			• List	• Evaluate	• Formulate
	• When			• Motive	• Explain	• Happen
	• Where			• Relationships	• Importance	• Imagine
	• Which			• Simplify	• Influence	• Improve
	• Who			• Survey	• Interpret	• Invent
	• Why			• Take part in	• Judge	• Make up
				• Test for	• Justify	• Maximize
				• Theme	• Mark	• Minimize
					• Measure	• Modify
					• Opinion	• Original
					• Perceive	• Originate
					• Prioritize	• Plan
					• Prove	• Predict
					• Rate	• Propose
					• Recommend	• Solution
					• Rule on	• Solve
					• Select	• Suppose
					• Support	• Test
					• Value	• Theory

Anderson, L. W., & Krathwohl, D. R. (2001). A taxonomy for learning, teaching, and assessing, Abridged Edition. Boston, MA: Allyn and Bacon.

Action Words for Bloom's Taxonomy

Knowledge	Understand	Apply	Analyze	Evaluate	Create
define	explain	solve	analyze	reframe	design
identify	describe	apply	compare	criticize	compose
describe	interpret	illustrate	classify	evaluate	create
label	paraphrase	modify	contrast	order	plan
list	summarize	use	distinguish	appraise	combine
name	classify	calculate	infer	judge	formulate
state	compare	change	separate	support	invent
match	differentiate	choose	explain	compare	hypothesize
recognize	discuss	demonstrate	select	decide	substitute
select	distinguish	discover	categorize	discriminate	write
examine	extend	experiment	connect	recommend	compile
locate	predict	relate	differentiate	summarize	construct
memorize	associate	show	discriminate	assess	develop
quote	contrast	sketch	divide	choose	generalize
recall	convert	complete	order	convince	integrate
reproduce	demonstrate	construct	point out	defend	modify
tabulate	estimate	dramatize	prioritize	estimate	organize
tell	express	interpret	subdivide	find errors	prepare
copy	identify	manipulate	survey	grade	produce
discover	indicate	paint	advertise	measure	rearrange
duplicate	infer	prepare	appraise	predict	rewrite
enumerate	relate	produce	break down	rank	role-play
listen	restate	report	calculate	score	adapt
observe	select	teach	conclude	select	anticipate
omit	translate	act	correlate	test	arrange
read	ask	administer	criticize	argue	assemble
recite	cite	articulate	deduce	conclude	choose
record	discover	chart	devise	consider	collaborate
repeat	generalize	collect	diagram	critique	collect
retell	give examples	compute	dissect	debate	devise
visualize	group	determine	estimate	distinguish	express
	illustrate	develop	evaluate	editorialize	facilitate
	judge	employ	experiment	justify	imagine
	observe	establish	focus	persuade	infer
	order	examine	illustrate	rate	intervene
	report	explain	organize	weigh	justify
	represent	interview	outline		make
	research	judge	plan		manage
	review	list	question		negotiate
	rewrite	operate	test		originate
	show	practice			propose
	trace	predict			reorganize
	transform	record			report
		schedule			revise
		simulate			schematize
		transfer			simulate
		write			solve
					speculate
					structure
					support
					test
					validate

4. SYLLABUS COPY AND SUGGESTED/REFERENCE BOOKS

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 BOOK:

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.

5. INDIVIDUAL TIME TABLE (B.MAMATHA)

Time	**9:10AM-10:10AM**	**10:10 AM-11:00AM**	**11:00 AM-11:50AM**	**11:50AM-12:40PM**	**12:40PM-1:20PM**	**1:20PM-2:20PM**	**2:20PM-3:10PM**	**3:10PM-4:00PM**
Period Day	**I**	**II**	**III**	**IV**		**V**	**VI**	**VII**
MON	**III D PPL**			**III B PPL**			**III B DAA LAB**	
TUE	**III D PPL**		**III B PPL**			**III D PPL**		
WED							**III D PPL**	
THU	**III B PPL**		**III A DAA LAB**				**III D PPL**	
FRI		**III B PPL**				**III B PPL**		
SAT		**III B PPL**						**III D PPL**

6. SESSION PLAN/LESSON PLAN

S.NO	Sub-Topic	NO. OF LECTURES REQUIRED	Planned Date	Conducted Date	Teaching Methods
UNIT-I PRELIMINARY CONCEPTS					
1	**Preliminary Concepts:** Reason for studying concepts of programming languages, Programming domains	**L1**	21/08/2023	24/08/2023	**M1,M2**
2	Language Evaluation Criteria	**L2**	25/08/2023	26/08/2023	**M2**
3	Influences on Language design,	**L3**	26/08/2023	31/08/2023	**M4**
4	Language categories, language design trade-offs	**L4**	1/09/2023	1/09/2023	**M1**
5	implementation methods, programming environments	**L5**	2/09/2023	4/09/2023	**M1,M4**
6	Evolution of Major Programming Languages.	**L6**	4/9/2023	7/09/2023	**M1,M4**
7	**Syntax and Semantics**	**L7**	7/09/2023	8/09/2023	**M1,M4, M19**
8	General problem of describing syntax	**L8**	8/09/2023	9/09/2023	**M1,M4, M19**
9	formal methods of describing syntax	**L9**	9/9/2023	11/09/2023	**M1,M4, M19**
10	attribute grammars	**L10**	11/09/2023	21/09/2023	**M1,M4, M19**
11	describing the meanings of programs	**L11**	14/09/2023	22/09/2023	**M1,M4**
UNIT 1 TEST			23/09/2023		
UNIT- II NAMES, BINDINGS AND SCOPE					
12	Introduction, names, variables	**L12**	16/09/2023	30/09/2023	**M1**
13	concept of binding, Scope, scope and lifetime referencing	**L13**	21/09/2023	3/10/2023	**M1,M4, M19**

	environments,				
14	named constants	**L14**	22/09/2023	6/10/2023	**M1,M4**
15	**Data types**: Introduction, primitive, character, string types, user defined ordinal types,	**L 15**	23/09/2023	9/10/2023	**M1,M4**
16	array , associative arrays, record, tuple types, list types, union types	**L 16**	25/09/2023	12/10/2023	**M1,M4**
17	Pointer and reference types, type equivalence	**L 17**	29/09/2023	13/10/2023	**M1,M4**
18	type checking, strong typing,	**L 18**	30/09/2023	13/10/2023	**M1,M4**
19	Arithmetic expressions, overloaded operators, type conversions	**L 19**	5/10/2023	14/10/2023	**M1,M4**
20	relational and boolean expressions, short-circuit evaluation,	**L 20**	6/10/2023	16/10/2023	**M1,M4**
21	assignment statements, mixed-mode assignment	**L 21**	7/10/2023	18/10/2023	**M1,M4**
22	**Control Structures** – introduction, selection statements	**L 22**	9/10/2023	19/10/2023	**M1,M4**
23	iterative statements , unconditional branching	**L 23**	12/10/2023	27/10/2023	**M1,M4**
24	guarded commands.	**L 24**	13/10/2023	27/10/2023	**M1,M4**
UNIT II TEST			28/10/2023		
UNIT-III SUBPROGRAMS					
25	Fundamentals of subprograms, design issues for subprograms	**L 25**	30/10/2023	4/11/2023	**M1,M4**

26	local referencing environments, parameter passing methods, parameters that are subprograms,	**L 26**	2/11/2023	6/11/2023	**M1,M4**
27	calling subprograms indirectly, overloaded subprograms generic subprograms, design issues for functions,	**L27,L28**	3/11/2023	10/11/2023 10/11/2023	**M1,M4**
28	user defined overloaded operators, closures, co routines, semantics calls	**L28,L29**	4/11/2023	11/11/2023 11/11/2023	**M1,M4**
29	stack-dynamic local variables, nested subprograms, blocks,	**L30,L31**	6/11/2023	16/11/2023 17/11/2023	**M1,M4**
30	dynamic scoping **Abstract Data types:** The concept of abstraction	**L 32**	9/11/2023	20/11/2023	**M1,M4**
31	introductions to data abstraction, encapsulation	**L 33**	10/11/2023	24/11/2023	**M1,M4**
32	constructs, naming encapsulations	**L 34**	11/11/2023	27/11/2023	**M1,M4**
UNIT-III TEST			**4/12/23**		
UNIT-IV OBJECT ORIENTED PROGRAMMING					
33	Design issues for OOP, OOP in Smalltalk, C++, Java, Ada 95.	**L 35**	13/11/2023	7/12/2023	**M1,M4**
34	OOP in Ruby, Implementation of Object-Oriented constructs	**L 36**	16/11/2023	8/12/2023	**M1,M4**
35	**Concurrency:** introduction,	**L 37**	17/11/2023	9/12/2023	**M1,M4**

	introduction to subprogram level, concurrency, message passing				
36	Languages,message passing	**L 38**	18/11/2023	11/12/2023	**M1,M4**
37	Ada support for concurrency, Java threads, functional languages	**L 39**	20/11/2023	13/12/2023	**M1,M4**
38	concurrency,Java threads, exception handling in java ,c++	**L 40**	23/11/2023	13/12/2023	**M1,M4**
39	**Event Handling:** Introduction,event handling in java	**L 41**	24/11/2023	14/12/2023	**M1,M4**
40	Event handling in C#	**L 42**	25/11/2023	14/12/2023/	**M1,M4**
41	Semaphores,monitors	**L 43**	27/11/2023	15/12/2023	**M1,M4**
42	Sub program level concurrency	**L 44**	30/11/2023	16/12/2023	**M1,M4**
43	Exception handling in Ada,java,C++	**L 45**	1/12/2023	16/12/2023	**M1,M4**
UNIT IVTEST			**23/12/2023**		
UNIT-V SCRIPTING LANGUAGES					
44	Introduction, mathematical functions, fundamentals of functional programming language. LISP, support for functions.	**L 46**	4/12/2023	16/12/2023	**M1,M4**
45	Pragmatics, Concepts of **Scripting Language, Python,Ruby.**	**L 47**	7/12/2023	18/12/2023	**M1,M4**
46	Python Data types,Values	**L 48**	8/12/2023	18/12/2023	**M1,M4**
47	Python Variables , Storage and Control	**L 49**	9/12/2023	18/12/2023	**M1,M4**
48	Bindings and	**L 50**	11/12/2023	19/12/2023	**M1,M4**

	Scope,binding types				
49	Data Abstraction in Python , Procedural Abstraction, Data Abstraction.	**L 51**	14/12/2023	19/12/2023	**M1,M4**
50	Binding in Python,Implicit and Explicit Binding, Compilation in Python	**L 52**	15/12/2023	19/12/2023	**M1,M4**
51	imperative languages,oops language, Introduction to an overview of logic programming	**L 53**	11/12/2023	20/12/2023	**M1,M4**
52	Basic elements of prolog,	**L 54**	14/12/2023	20/12/2023	**M1,M4**
53	Module Library,oops in python deficiencies of prolog, applications of logic programming	**L 55**	15/12/2023		**M1,M4**
54	Procedural Abstraction, Data Abstraction. Binding in Python, Implicit and Explicit Binding	**L 56**	16/12/2023	20/12/2023	**M1,M4**
UNIT V TEST 23/12/2023					
TOTAL			**56**		

METHODS OF TEACHING:

M1:Lecture Method	**M11:Tutorial**
M2:Demo Method	**M12:Assignment**
M3:Guest Lecture	**M13:Industry Visit**
M4:Presentation/PPT	**M14:Project Based Learning**
M5:Mind Map	**M15:Mnemonics**
M6:ATL Lab	**M16:Laboratory Improvement Future Trends**
M7:Group Learning	**M17:Collaborative Learning**
M8:One minute Paper	**M18:Think Pair Share**
M9 :Case Study	**M19:NPTEL Video Lectures**
M10:Flipped Classes	**M20:Innovative Assignment**

7. SESSION EXECUTION LOG:

S no	Unit	Scheduled started date	Completed date	Remarks
1	I	24/08/23	22/09/23	COMPLETED
2	II	30/09/23	27/10/23	COMPLETED
3	III	04/11/23	27/11/23	COMPLETED
4	IV	07/12/23	18/12/23	COMPLETED
5	V	16/12/23	20/12/23	COMPLETED

8.LECTURE NOTES – (HAND WRITTEN)

9.ASSIGNMENT QUESTIONS ALONG WITH SAMPLE ASSIGNMENTS SCRIPTS

CMR ENGINEERING COLLEGE
EXPLORE TO INVENT

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Approved by AICTE - New Delhi. Affiliated to JNTUH and Accredited by NAAC & NBA)

Kandlakoya (V), Medchal (M), Medchal - Malkajgiri (D)-501401

NATIONAL BOARD OF ACCREDITATION

NAAC

PPL-MID-1 ASSIGNMENT-1

1. a) **Describe** the steps involved in the language evaluation criteria.[CO1]

   b) **Explain** the various potential benefits of studying programming language concepts.[CO1]

2. a) **What** are various programming domain and implementation methods of

   Programming language.[CO2]

3. a) **Explain** about attribute grammars with example.[CO1]

   b) **Explain** the process of compilation in each phase of a compiler.[CO1]

4. **Explain** the following data types: Union, Pointer and Reference.[CO2]

5. a) **Explain** Arithmetic expressions, Boolean expressions and relational expressions in

   various programming languages? [CO2]

   b) **Distinguish** between data types :arrays and records.[CO2]

PPL-MID-2 ASSIGNMENT-2

1. a) **Explain** the overloaded subprogram with an examples.(CO3)

   b) **List** and explain different design issues for subprogram.(CO3)

2. a) **What** is abstraction? What are parameterized abstract data types? (CO3)

   b) **Describe** different parameter passing methods with an example. (CO3)

3. a) **What** is event? Write about event handling in java with example. (CO4)

   b) **Explain** how to handle the exception in C++.(CO4)

4. **Explain** how semaphores can be used to implement concurrency.(CO4)

5. a) **Write** about fundamentals of functional programming language.(CO5)

   b) **What** are basic elements of PROLOG? (CO5)

10. MID EXAM QUESTION PAPERS ALONG WITH SAMPLE ANSWERS SCRIPTS

CMR ENGINEERING COLLEGE
EXPLORE TO INVENT

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Approved by AICTE - New Delhi. Affiliated to JNTUH and Accredited by NAAC & NBA)

Kandlakoya (V), Medchal (M), Medchal - Malkajgiri (D)-501401

NBA NATIONAL BOARD OF ACCREDITATION

III-B.TECH I -SEM- I-MID EXAMINATIONS

Date: 3-11-2023 **Time:10:00 AM to11:30 AM**

Subject: PPL[B, D- Section] Branch: CSE **Max.Marks:25 M**

Note: Question paper contains two parts, Part-A and Part- B.

Part-A is compulsory which carries 10 marks.

Answer all questions in part-A.

Answer anyone full question from each unit. Each question carries5 marks.

PART-A **5 x 2M = 10 M**

		BTL	CO
1.	**What** is type checking?	1	2
2.	**What** is a named constant?	1	2
3.	**Define** precedence and associativity of operators.	1	1
4.	**Explain** about short-circuit evaluation.	2	2
5.	**Define** binding and lifetime.	1	2

PART-B **3 x 5M = 15 M**

		BTL	CO
6.	**Explain** about language evaluation criteria.	2	1
	(OR)		
7.	**What** is hybrid implementation? **Explain** with a neat flow diagram.	1,2	1
8.	Consider the following grammar	2	1

<S> -> <A> a <B> b

<A> -> <A> b | b

<B> -> b

Check whether the following sentence is generated by this grammar.

i)babb ii)Bbbabb

(OR) **2** **2**

9. **Explain** about selection and iterative statements.

10. **Define** Coercion,type error,type checking and strong Typing. **1** **2**

(OR)

11. A)**Explain** in brief about pointer and reference types **2** **2**

B)**Explain** about concept of binding. **2** **2**

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Approved by AICTE - New Delhi. Affiliated to JNTUH and Accredited by NAAC & NBA)

Kandlakoya (V), Medchal (M), Medchal - Malkajgiri (D)-501401

III-B.TECH I -SEM- II-MID EXAMINATIONS

Date: 30-12-2023 **Time: 10:00 AM to11:30 AM**

Subject: PPL [B, D- Section] Branch: CSE **Max.Marks:25 M**

Note: Question paper contains two parts, Part-A and Part- B.

Part-A is compulsory which carries 10 marks.

Answer all questions in part-A.

Answer anyone full question from each unit. Each question carries5 marks.

PART-A **5 x 2M = 10 M**

		BTL	CO
1.	**Define** Abstract Data types.	1	3
2.	**What** are the design issues of exception handling?	1	4
3.	**What** are the four possible levels of concurrency in programs?	1	4
4.	**Distinguish** between checked and unchecked exception.	4	4
5.	**Describe** the syntax and semantics of map car.	4	4

PART-B **3 x 5M = 15 M**

		BTL	CO
6.	**Write** about implementation of subprograms with stack-dynamic local variables.	6	3
	(OR)		
7.	**Explain** different parameter passing methods and parameters that are subprograms.	2	3
8.	**Discuss** exception handling in java and Ada language.	6	4
	(OR)		
9.	**Explain** how semaphores can be used to implement concurrency.	2	4

10.	a) **What** are the application of Functional Language.	**1**	**5**
	b) **Explain** various control statements available in python.	**5**	**5**
	(OR)		
11.	a) **Compare** Functional and imperative languages.	**4**	**5**
	b) **Explain** the basic elements of prolog in detail.	**2**	**5**

11 .SCHEME OF EVALUATION

MID-1-SCHEME OF EVALUATION-

PART-A

SNO	THEORY	MARKS	TOTAL
1	**What** is type checking?	2	2
2	**What** is a named constant?	2	2
3	**Define** precedence and associativity of operators.	2	2
4	**Explain** about short-circuit evaluation.	2	2
5	**Define** binding and lifetime.	2	2

PART-B

S.NO	THEORY	MARKS	TOTAL
6	**Explain** about language evaluation criteria.	5	5
7	**What** is hybrid implementation? **Explain** with a neat flow diagram.	5	5
8	Consider the following grammar <S> -> <A> a <B> b <A> -> <A> b | b <B> -> b Check whether the following sentence is generated by this grammar. i)babb ii)Bbbabb	5	5
9	**Explain** about selection and iterative statements.	5	5
10	**Define** Coercion, type error, type checking and strong Typing.	5	5
11.a	**Explain** in brief about pointer and reference types.	3	3
11.b	**Explain** about concept of binding.	2	2
TOTAL MARKS		**15**	**15**

MID-2-SCHEME OF EVALUATION

PART-A

SNO	THEORY	MARKS	TOTAL
1	**Define** Abstract Data types.	2	2
2	**What** are the design issues of exception handling?	2	2
3	**What** are the four possible levels of concurrency in programs?	2	2
4	**Distinguish** between checked and unchecked exception.	2	2
5	**Describe** the syntax and semantics of map car.	2	2
	TOTAL		**10**

PART-B

S.NO	**THEORY**	**MARKS**	**TOTAL**
6	**Write** about implementation of subprograms with stack-dynamic local variables.	5	5
7	**Explain** different parameter passing methods and parameters that are subprograms.	5	5
8	**Discuss** exception handling in java and Ada language.	5	5
9	**Explain** how semaphores can be used to implement concurrency.	5	5
10.a	**What** are the application of Functional Language.	2	2
10.b	**Explain** various control statements available in python.	3	3
11.a	**Compare** Functional and imperative languages.	2	2
11.b	**Explain** the basic elements of prolog in detail.	3	3
TOTAL MARKS		**15**	**15**

12.Mapping of Course Objectives, Course Outcomes with PEOs and Pos

	Program Outcome(PO):												
	PO	PO1	PO2	PO3	PO4	PO 5	PO6	PO7	PO8	PO 9	PO10	PO11	PO12
PEOs		X	X		X								
			X					X				X	
					X			X		X	X		X
		X									X		

Course Outcomes	Relationship of Course outcomes to Program Outcomes (PO AVG)												
CO-PO&PSO MATRIX	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1	3	3	-	-	2	-	-	-	-	-	-	-	2	-
CO2	-	-	2	-	2	-	2	-	-	-	-	-	2	-
CO3	2	2	3	-	3	-	-	-	-	-	-	-	-	1
CO4	3	3	2	-	-	-	-	-	-	-	-	2	-	1
CO5	-	-	2	-	3	-	2	-	-	-	-	-	-	-
AVERAGE	**3**	**3**	**2**	**0**	**3**	**0**	**2**	**0**	**0**	**0**	**0**	**2**	**2**	**1**

13.COs,POs,PSOs JUSTIFICATION :

COURSE OUTCOMES

CO1	**Explain** the important features of the Programming Languages.
CO2	**Compare** different Programming Domains.
CO3	**Evaluate** Merits and Demerits of a Particular Programming Language.
CO4	**Choose** Specific Programming Language for the **Development** of Specific Applications.
CO5	Understand and **Analyze** the Importance of Implementation Process.

COURSE	**Relationship of Course outcomes to Program Outcomes (PO AVG)**													
CO-PO&PSO MATRIX	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**
CO1	3	3	-	-	2	-	-	-	-	-	-	-	2	-
CO2	-	-	2	-	2	-	2	-	-	-	-	-	2	-
CO3	2	2	3	-	3	-	-	-	-	-	-	-	-	1
CO4	3	3	2	-	-	-	-	-	-	-	-	2	-	1
CO5	-	-	2	-	3	-	2	-	-	-	-	-	-	-

Justification:

CO1: Able to Explain the important features of the Programming Languages
Correlated with PO1 High: Because it contributes the knowledge on Principles of Programming Languages and student can understand Preliminary concepts and syntax and semantics of different Programming Languages. So, overall the correlation of CO1 to PO1 is good.
Correlated with PO2 High: as this course outcome provides students to know about Language Categories and Language Design Trade-offs. So, overall the correlation of CO2 is good.
Correlated with PO5 moderately: It contributes to identify the problems that arises but formal methods of describing syntax is moderate. So, overall the correlation of CO1 is weak.
Correlated with PSO1 moderately: Because it contributes the knowledge on Principles of Programming Languages and student can know how to describe the Meaning of the Programs.

CO2:. Able to Compare different Programming Domains

Correlated with PO3 moderately: Because it provides fundamentals of computer science and students know about names,bindings, Data Types and also Expressions and Statements and Control structures. So, correlation is Moderate.
Correlated with PO5 moderately: contribution to provide solutions for Different Arithmetic operators,Overloaded Operator , the correlation is moderate.
Correlated with PO7 moderately: It contributes to provide scope of variables,identifying some solution to complex problems but not a complete. So, the correlation of CO2 is moderate.
Correlated with PSO1 moderately: Because it contributes the knowledge on Principles of Programming Languages and student can know about iterative statements,Selection statements . So, the correlation of CO2 is moderate.
CO3:. Ability to Evaluate Merits and Demerits of a Particular Programming Language
Correlated with PO1moderately: contribution of this course outcome is weak for providing solutions for complex problems i.e in research area. The correlation is moderate.
Correlated with PO2 moderately: the CO contributes knowledge on different techniques on message passing among processes such that the student gets knowledge on using modern tools. So, the correlation of CO is Good.
Correlated with PO3 moderately: Students get knowledge on different techniques of message passing so that it motivates student to learn new technologies. The correlation is moderate
Correlated with PO5 moderately: contribution to provide solutions for Different Abstract data types,and impelementing subPrograms ,so the correlation is moderate.
Correlated with PSO1 moderately: Because it contributes the knowledge on Principles of Programming Languages and student can know how to describe the Meaning of the Programs. So, the correlation of CO2 is moderate.

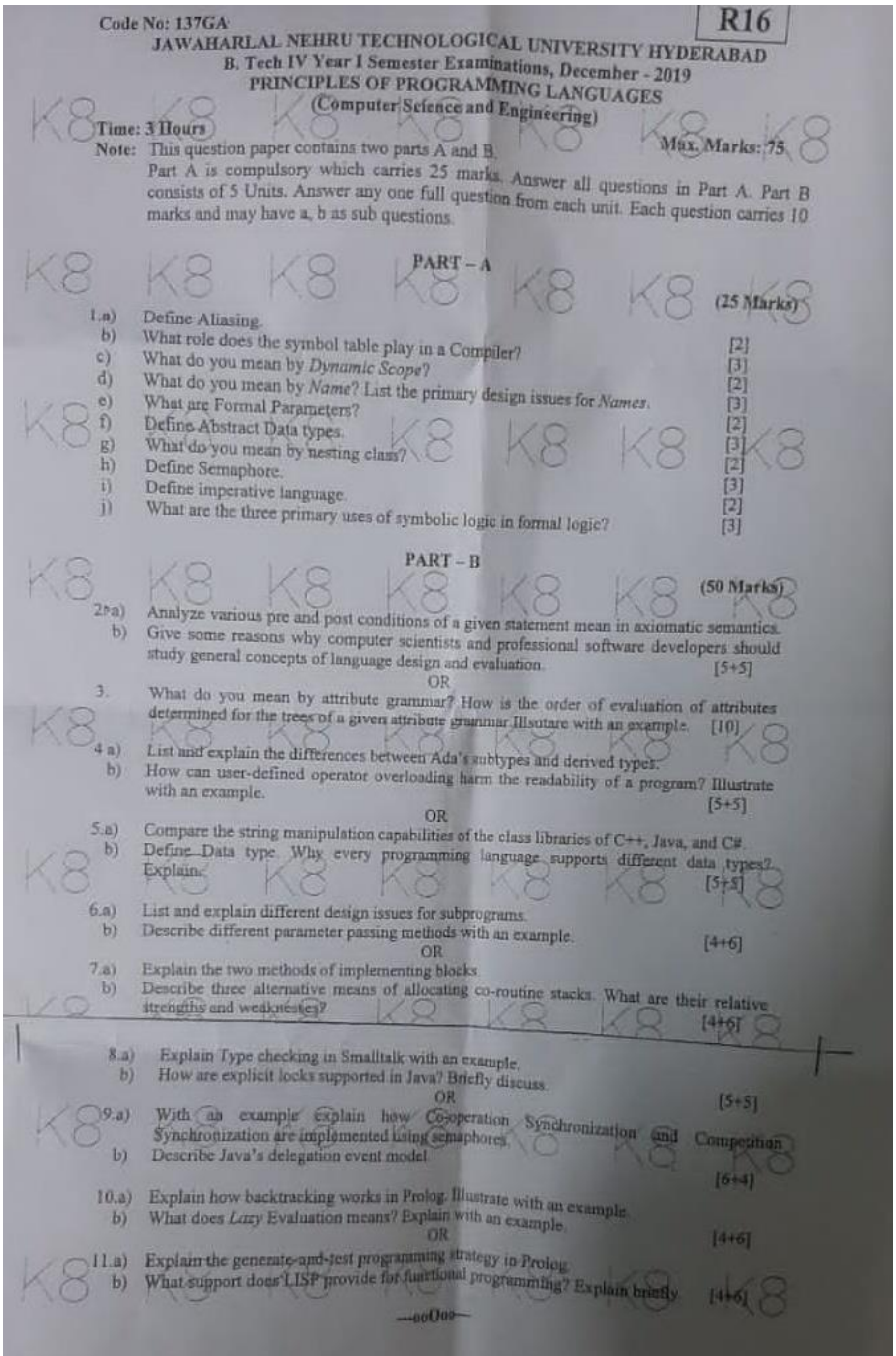
CO4: Able to Choose Specific Programming Language for the Development of Specific Applications
Correlated with PO1 high: Because it contributes the knowledge on fundamentals of object oriented Programming which makes students get engineering knowledge and student can categorize different concurrency. So, overall the correlation of CO4 to PO1 is good.
Correlated with PO2 high: as this course outcome provides students identify different Exceptional Handling Problems that occur when dealing with processes. So, overall the correlation of CO1 is good.
Correlated with PO3 moderately: It contributes only knowledge on developing complex problems but, cannot provide a complete solution to Complex problems. So, overall the

correlation of CO4 is good.
Correlated with PO12 moderately: It contributes only knowledge on developing complex problems but, cannot provide a complete solution to Complex problems. So, overall the correlation of CO4 is good.
Correlated with PSO2 low: Because it contributes the knowledge on Principles of Programming Languages and student can know how to describe the Meaning of the Programs but, cannot provide a complete solution to Complex problems..
CO5: . Able to Understand and Analyze the Importance of Implementation Process
Correlated with PO3 moderately: Because it contributes the knowledge on fundamentals of Functional Programming which makes students get engineering knowledge and student can get knowledge on Logic ProgrammingLanguages. So, overall the correlation of CO1 to PO1 is Moderate.
Correlated with PO5 moderately: as this course outcome provides students identify different Problems that occur when dealing with processes but cannot provide better solution for solving the issues So, overall the correlation of CO1 is Moderate.
Correlated with PO7 High: outcome contributes better for identification of different solutions for problems. So, that the students can apply to build some applications. So the correlation is high.

14. ATTAINMENT OF CO's, PO's AND PSO's (EXCELSHEET):

RESULT NOT RELEASED .

15. PREVIOUS YEAR QUESTION PAPERS



Code No: 137GA **R16**

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD
B. Tech IV Year I Semester Examinations, December - 2019
PRINCIPLES OF PROGRAMMING LANGUAGES
(Computer Science and Engineering)

Time: 3 Hours **Max. Marks: 75**

Note: This question paper contains two parts A and B.
Part A is compulsory which carries 25 marks. Answer all questions in Part A. Part B consists of 5 Units. Answer any one full question from each unit. Each question carries 10 marks and may have a, b as sub questions.

PART – A

(25 Marks)

1.a) Define Aliasing. [2]
b) What role does the symbol table play in a Compiler? [3]
c) What do you mean by *Dynamic Scope*? [2]
d) What do you mean by *Name*? List the primary design issues for *Names*. [3]
e) What are Formal Parameters? [2]
f) Define Abstract Data types. [3]
g) What do you mean by nesting class? [2]
h) Define Semaphore. [3]
i) Define imperative language. [2]
j) What are the three primary uses of symbolic logic in formal logic? [3]

PART – B

(50 Marks)

2.a) Analyze various pre and post conditions of a given statement mean in axiomatic semantics.
b) Give some reasons why computer scientists and professional software developers should study general concepts of language design and evaluation. [5+5]

OR

3. What do you mean by attribute grammar? How is the order of evaluation of attributes determined for the trees of a given attribute grammar Illsutare with an example. [10]

4 a) List and explain the differences between Ada's subtypes and derived types.
b) How can user-defined operator overloading harm the readability of a program? Illustrate with an example. [5+5]

OR

5.a) Compare the string manipulation capabilities of the class libraries of C++, Java, and C#.
b) Define Data type. Why every programming language supports different data types? Explain. [5+5]

6.a) List and explain different design issues for subprograms.
b) Describe different parameter passing methods with an example. [4+6]

OR

7.a) Explain the two methods of implementing blocks.
b) Describe three alternative means of allocating co-routine stacks. What are their relative strengths and weaknesses? [4+6]

8.a) Explain Type checking in Smalltalk with an example.
b) How are explicit locks supported in Java? Briefly discuss. [5+5]

OR

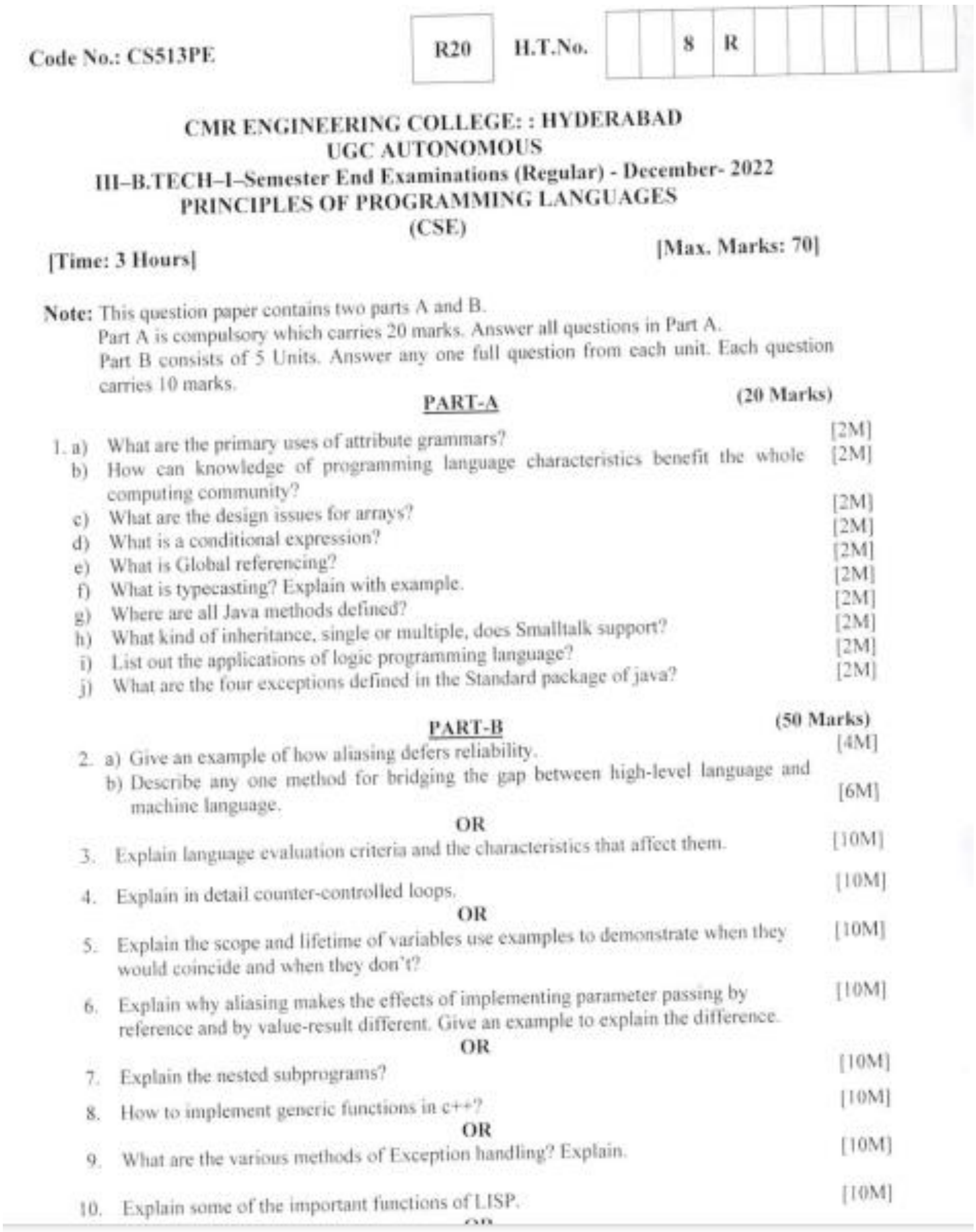
9.a) With an example explain how Co-operation Synchronization and Competition Synchronization are implemented using semaphores.
b) Describe Java's delegation event model. [6+4]

10.a) Explain how backtracking works in Prolog. Illustrate with an example.
b) What does *Lazy* Evaluation means? Explain with an example. [4+6]

OR

11.a) Explain the generate-and-test programming strategy in Prolog.
b) What support does LISP provide for functional programming? Explain briefly. [4+6]

---ooOoo---



Code No.: CS513PE

R20	H.T.No.			8	R						

CMR ENGINEERING COLLEGE: : HYDERABAD
UGC AUTONOMOUS

III–B.TECH–I–Semester End Examinations (Regular) - December- 2022

PRINCIPLES OF PROGRAMMING LANGUAGES

(CSE)

[Time: 3 Hours] [Max. Marks: 70]

Note: This question paper contains two parts A and B.
Part A is compulsory which carries 20 marks. Answer all questions in Part A.
Part B consists of 5 Units. Answer any one full question from each unit. Each question carries 10 marks.

PART-A (20 Marks)

1. a) What are the primary uses of attribute grammars? [2M]
   b) How can knowledge of programming language characteristics benefit the whole computing community? [2M]
   c) What are the design issues for arrays? [2M]
   d) What is a conditional expression? [2M]
   e) What is Global referencing? [2M]
   f) What is typecasting? Explain with example. [2M]
   g) Where are all Java methods defined? [2M]
   h) What kind of inheritance, single or multiple, does Smalltalk support? [2M]
   i) List out the applications of logic programming language? [2M]
   j) What are the four exceptions defined in the Standard package of java? [2M]

PART-B (50 Marks)

2. a) Give an example of how aliasing defers reliability. [4M]
   b) Describe any one method for bridging the gap between high-level language and machine language. [6M]

OR

3. Explain language evaluation criteria and the characteristics that affect them. [10M]

4. Explain in detail counter-controlled loops. [10M]

OR

5. Explain the scope and lifetime of variables use examples to demonstrate when they would coincide and when they don't? [10M]

6. Explain why aliasing makes the effects of implementing parameter passing by reference and by value-result different. Give an example to explain the difference. [10M]

OR

7. Explain the nested subprograms? [10M]

8. How to implement generic functions in c++? [10M]

OR

9. What are the various methods of Exception handling? Explain. [10M]

10. Explain some of the important functions of LISP. [10M]

R16

Code No: 137GA

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

B. Tech IV Year I Semester Examinations, January/February - 2023

PRINCIPLES OF PROGRAMMING LANGUAGES

(Computer Science and Engineering)

Time: 3 Hours **Max. Marks: 75**

Note: i) Question paper consists of Part A, Part B.
ii) Part A is compulsory, which carries 25 marks. In Part A, Answer all questions.
iii) In Part B, Answer any one question from each unit. Each question carries 10 marks and may have a, b as sub questions.

PART – A

(25 Marks)

1.a)	Why language evaluation criteria need to be considered?	[2]
b)	Compare Syntax and Semantic of a program.	[3]
c)	Define binding and lifetime.	[2]
d)	Differentiate Relational and Boolean Expressions.	[3]
e)	Discuss parameter passing methods.	[2]
f)	Briefly specify about overloaded operators.	[3]
g)	What are the various issues of OOP?	[2]
h)	Discuss concurrency in functional languages.	[3]
i)	Explain scope and binding in python.	[2]
j)	Compare procedural abstraction and data abstraction.	[3]

PART – B

(50 Marks)

2.a) Discuss formal method for describing syntax using BNF.
b) What are various programming domain? Explain. [5+5]

OR

3.a) Give EBNF description for the 'C' Union.
b) How are imperative languages built? [5+5]

4.a) Compare static and dynamic type checking.
b) Give an overview of iterative statements. [5+5]

OR

5.a) Discuss the advantages and disadvantages of the interoperability of pointers and arrays in C.
b) Write a note on type equivalence. [5+5]

6.a) Discuss Encapsulation in detail.

b) Consider the following program.

```
        (define A
           (lambda()
              (let*((x2)
                   (C  (lambda (p)
                          (let  ((x 4))
                           (p))))
                  (D  (lambda ()
                          (let  ((x 3))
                            (C D)))))
            (B))))
```

What does this program print? What would it print if used dynamic scoping. [5+5]

OR

7.a) Explain general semantics of call and return statements.

b) Mention the advantages and disadvantages of Abstract data types. [5+5]

8.a) Highlight the design issues in C++ and Java.

b) What are the various features of RUBY programming language? [5+5]

OR

9.a) Discuss exceptional handling in ADA.

b) What is a monitor? How do monitor condition variables differ from those of semaphores? [5+5]

10.a) Explain the different type variables supported in PYTHON .

b) Distinguish Functional and Imperative programming language. [5+5]

OR

11.a) Discuss values and types in Python language.

b) Explain different mathematical functions supported in LISP language. [5+5]

---ooOoo---

UNIT WISE IMPORTANT QUESTIONS

Unit-I

1. What arguments can you make for the idea of a single language for all programming domains?
2. Explain the different aspects of cost of a programming language
3. Discuss the major features that a perfect programming language includes in your opinion.
4. Describe briefly the advantages and disadvantages of some programming environments you have used.
5. Write an evaluation of some programming language you know using the criteria for a good programming language.
6. Discuss about the programming paradigms with example.
7. Give some reasons why computer scientists and professional software developers should study general concepts of programming languages.
8. Briefly discuss a few of the areas of computer applications and their associated languages.
9. a) Give an example of how aliasing defers reliability.
10. (b) Explain with examples how syntactic design choices affect readability
11. What are the three general methods of implementing a programming language Explain briefly
12. What are the reasons for studying the concepts of Programming Language?
13. Explain about Language Evolution Criteria
14. What are the various influences on Language Design?
15. Explain various Implementation methods of Programming language?

Unit-II

1. Write a simple assignment statement with one arithmetic operator in some language you Know. For each component in the statement, list the various bindings that are required to determine the semantics when the statement is executed. For each binding, indicate the binding time used for the language.
2. Dynamic type binding is closely related to implicit heap-dynamic variables. Explain this relationship.
3. What disadvantages are there in implicit dereferencing of pointers, but only in certain Contexts? For example, consider the implicit dereference of a pointer to a record in Ada When it is used to reference a record field.
4. What significant justification is there for the → operator in C and C++?

5. What are the arguments for the inclusion of enumeration types in C#, although they are not in Java?
6. Explain in detail arrays, indices, subscript bindings and array categories.
   (a) What are the advantages and disadvantages of static variables?
   (b) How can enumeration types can be implemented in languages that do not have enumeration types and what are the problems associated with such implementations?
7. Explain pointers in C and C++.?
8. What is meant by binding? Discuss the various types of binding with examples.
9. Explain in detail various design issues of character string types.
10. Explain stack dynamic variables & Explicit heap dynamic variables?
11. Explain heap management of a single & variable size segments?
12. Explain about various Array categories?
13. Write on decimal data types? Write about advantages & disadvantages?
14. What are the design issues for string length & enumeration types?
15. Define static, fixed-stack dynamic, stack-dynamic, stack-dynamic, fixed-heap dynamic, heap-dynamic arrays? What are the advantages on each?
16. What are the arguments for the inclusion of enumeration types in C#, although they are not in Java?

UNIT III:

1. What are different parameter passing methods? Discuss with examples.
2. Explain about genetic subroutines and modules.
3. Explain type checking techniques in parameter passing.
4. Explain how multidimensional arrays are passed as parameters
5. What are the three semantic models of parameter passing?
6. What are the different dynamic referencing environments of subprograms
7. Explain why it may sometimes be useful for a function to have side effects
8. Describe the parameter modes of Ada. How do they differ from the modes of most other Algol-family languages
9. Explain how subprograms names are passed as parameters.
10. What are general characteristics of subprograms?
11. Explain the categories of subprograms with suitable example
12. What are the design issues of subprograms? explain
13. What is co-routine ? explain with an example.
14. Explain how subprogram is overloaded. Give examples.
15. Distinguish between actual parameters and formal parameter methods
16. Discuss how generic functions are implemented in C++.
17. Explain in detail the local referencing environment of a sub-program
18. Explain three semantic models of parameter passing
19. What is Aliasing? Discuss the issues involved in aliasing parameters in procedures?
20. Explain all of the difference between subtypes and derived types.
21. In what different places can the definition of a C++ member function appears?

22. Discuss how generic functions are implemented in java.
23. What is abstraction?What are design issues of data abstraction?What are parameterized abstract data types?
24. Explain about parameterized ADT.

UNIT IV:

1. Explain about object oriented programming? How it is different from functional Programming?
2. Discuss about the concurrency control with examples.
3. Explain about the message passing mechanism
4. How exception handling is done in java? Explain with example.
5. What are the primary problems with using semaphores to provide synchronization
6. What are different types of inheritance ?explain with examples
7. Explain about data abstraction and encapsulation
8. What are the basic concepts of exception handling and what are the design issues of exception handaling.

9.Explain in detail about the following concepts in Object Oriented Programming:
 i) Encapsulation
 ii) Inheritance
 iii) Dynamic Method Binding

11. Explain in detail about various concepts in Object Oriented Programming.

12. Name three important benefits of abstraction
13. What are private and limited private types in Ada?
14. Describe Predicate functions for symbolic atoms and list.
15. What is a metaclass in Smalltalk? Explain the difference between initialization and assignment in C++ with suitable examples.
16. What is a monitor? Explain usage of monitors with example in concurrent Pascal to implement cooperation synchronization
17. Distinguish between checked and unchecked exceptions.

18.Explain about threads in Java and C #.

19. Compare the exception handling facilities of C++ with these of Ada.
20. What are the categories of concurrency ?What is the need for studying concurrency
21. What is task synchronization? Explain in detail two kinds of task synchronization.
22.What are the design issues of exception handling
23. What is a thread? Explain how threads are implemented in java.
24.Discuss in detail how the encapsulation and data abstraction is achieved in Java with suitable exam ples
25. What is competition synchronization? Explain the need for competition synchronization
26. What causes a C++ template function to be instantiated?
27. Write an analysis of the similarities of and differences between java packages and C++ namespaces?
28. What are the language design issues for abstract data types?explain

29. Explain the difference between physical and logical concurrency?
30. What is binary semaphore ? Explain counting semaphore?
31. Discuss the design issues for concurrency
32. Compare procedure oriented and object oriented programming. Explain the object oriented features supported by C++.
33. Define semaphores. Explain, how cooperation synchronization and competition synchronization are implemented using semaphores.

UNIT V:

1. Discuss about basic elements of Prolog.
2. Explain in detail about Concurrent programming fundamentals.
3. What are the applications of Logic Programming?
4. Explain Logic programming concepts in detail

5. Describe the difference between forward chaining and backward chaining. Which chaining in used in PROLOG by default
6. What is meant by logic programming? Explain the uses of symbolic logic in formal logic.
7.Explain about the origins of PROLOG

8.Explain the following

A)meta language
B) logic programming
C) prolog

9. Explain about functional forms provided in LISP.

10 . Explain Functional Programming concepts in detail.

11 . What are the features provided by the functional languages over imperative languages? Explain each with suitable example.
12 . Discuss in detail about the different data structures that are present in LISP with suitable examples
13 . Give brief description about the dialects of LISP.

14. What is a package? Explain them with respect to python language.
15 . Discuss in detail about the various data types and structures that are present in LISP.
16 . Write procedure to find the greatest common divisor by using python language
17. With the help of a suitable example, explain the modules using python
18. What are the fundamentals of functional programming languages.
19. Explain various operations that can be performed on atoms and lists in LISP. Give examples.

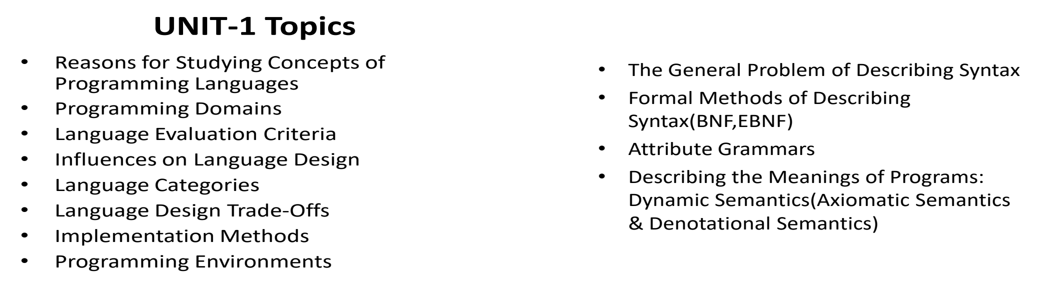
20.What are the difference between CONS, LIST and APPENED?
21. Write a LISP function Fib(n) that computes nth Fibonacci number.
22. What scoping rule is used in
a) COMMON LISP

b) ML

c) Haskell.

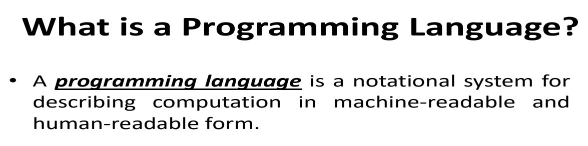
23. Write a detail note on functions in ML.
24. Give comparison of Functional and Imperative Languages.
25. Write about the following.
    a) logic programming languages.
    b) Scripting languages.
    c) Functional programming languages.
26. Write a function that computes the sum of numbers using vectors in LISP.

16. POWER POINT PRESENTATIONS: (SOFTCOPY)



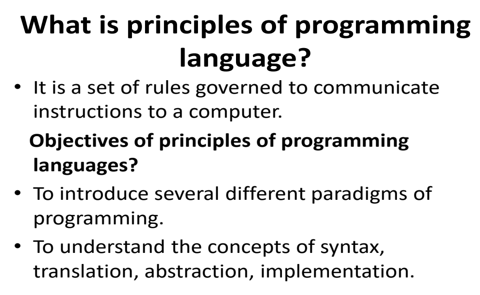
UNIT-1 Topics

- 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
- The General Problem of Describing Syntax
- Formal Methods of Describing Syntax(BNF,EBNF)
- Attribute Grammars
- Describing the Meanings of Programs: Dynamic Semantics(Axiomatic Semantics & Denotational Semantics)



What is a Programming Language?

- A ***programming language*** is a notational system for describing computation in machine-readable and human-readable form.

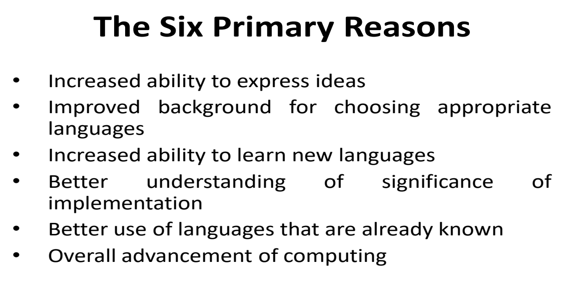


What is principles of programming language?

- It is a set of rules governed to communicate instructions to a computer.

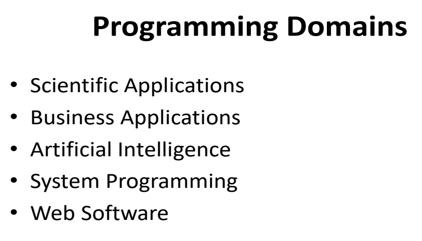
Objectives of principles of programming languages?

- To introduce several different paradigms of programming.
- To understand the concepts of syntax, translation, abstraction, implementation.



The Six Primary Reasons

- Increased ability to express ideas
- Improved background for choosing appropriate languages
- Increased ability to learn new languages
- Better understanding of significance of implementation
- Better use of languages that are already known
- Overall advancement of computing



Programming Domains

- Scientific Applications
- Business Applications
- Artificial Intelligence
- System Programming
- Web Software

Language Evaluation Criteria

- **Readability** – the ease with which programs can be read and understood.
- **Writability** – the ease with which programs can be developed for a given program domain.
- **Reliability** – the extent to which a program will perform according to its specifications.
- **Cost** – the ultimate total cost.

Influences on Language Design

- The important factors that influence the basic design of Programming languages are:
- Computer Architecture
- Programming Methodologies

Programming Methodologies

- 1950s and early 1960s: Simple applications; worry about machine efficiency
- Late 1960s: machine efficiency became important; readability, better control structures
  - structured programming
  - top-down design and step-wise refinement
- Late 1970s: Process-oriented to data-oriented
  - data abstraction
- The first language was SIMULA 67
- Middle 1980s: Object-oriented programming
  - Data abstraction + inheritance + polymorphism

Example: Smalltalk,Ada 95,C++,Java

Language Categories

- There are four different programming language paradigms:
  - Imperative
  - Functional
  - Logic
  - Object-Oriented

Language Design Trade-Offs

- **Reliability vs. cost of execution**

  Example: Java demands all references to array elements be checked for proper indexing, which leads to increased execution costs
- **Readability vs. writability**

  Example: APL provides many powerful operators (and a large number of new symbols), allowing complex computations to be written in a compact program but at the cost of poor readability
- **Writability (flexibility) vs. reliability**

  Example: C++ pointers are powerful and very flexible but are unreliable

Implementation Methods

1. Compilation
2. Pure Interpretation
3. Hybrid Implementation Systems

The General Problem of Describing Syntax

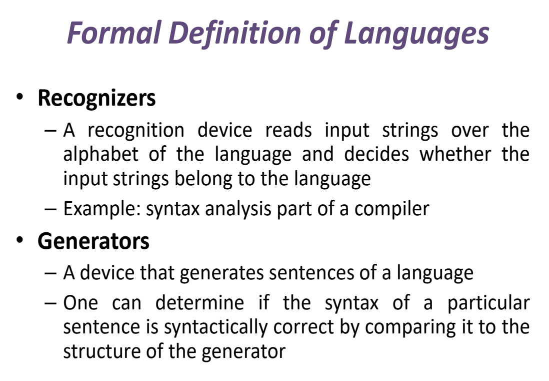
Terminology

- A *sentence* is a string of characters over some alphabet
- A *language* is a set of sentences
- A *lexeme* is the lowest level syntactic unit of a language (e.g., *, sum, begin)
- A *token* is a category of lexemes (e.g., identifier)

- For example, consider the following java statement:

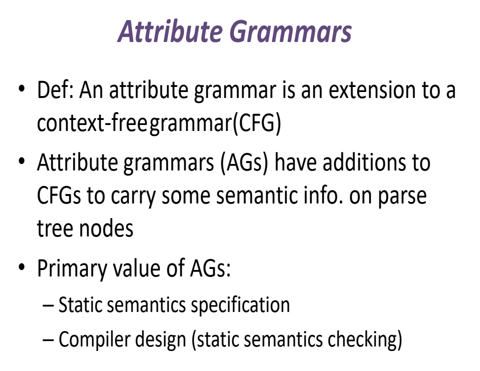
  index = 2 * count + 17;
- The lexemes and tokens of this statement are :

Lexemes	**Tokens**
index	identifier
=	equal_sign
2	int_literal
*	mult_op
count	identifier
+	plus_op
17	int_literal
;	semicolon



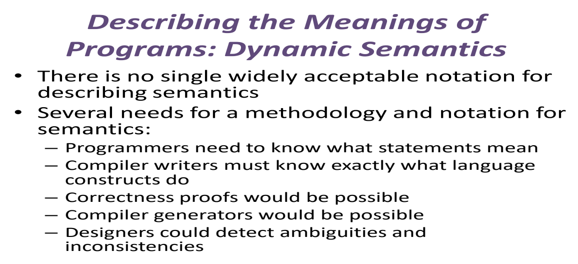
Formal Definition of Languages

- **Recognizers**
  - A recognition device reads input strings over the alphabet of the language and decides whether the input strings belong to the language
  - Example: syntax analysis part of a compiler
- **Generators**
  - A device that generates sentences of a language
  - One can determine if the syntax of a particular sentence is syntactically correct by comparing it to the structure of the generator



Attribute Grammars

- Def: An attribute grammar is an extension to a context-freegrammar(CFG)
- Attribute grammars (AGs) have additions to CFGs to carry some semantic info. on parse tree nodes
- Primary value of AGs:
  - Static semantics specification
  - Compiler design (static semantics checking)



Describing the Meanings of Programs: Dynamic Semantics

- There is no single widely acceptable notation for describing semantics
- Several needs for a methodology and notation for semantics:
  - Programmers need to know what statements mean
  - Compiler writers must know exactly what language constructs do
  - Correctness proofs would be possible
  - Compiler generators would be possible
  - Designers could detect ambiguities and inconsistencies

Thank You

17. INNOVATIVE TEACHING METHODS IF ANY (ATTACHED INNOVATIVE ASSIGNMENT)

1. Give some reasons why computer scientist and professional software developers should study general concepts of language design and evaluation?
2. Write reasons for the statement. “Exception handling is very important, but often neglected by programming languages?

18. REFERENCES(Text book/websites/journals)

- https://nptel.ac.in/courses/106/102/106102067/
- https://he.kendallhunt.com/sites/default/files/uploadedFiles/Kendall_Hunt/Content/Higher Education/Uploads/ChenTsai_ProgramLanguages_4e_Chapter1.pdf
- https://www.slideshare.net/krishnasai90663/principles-ofprogramminglanguageslecturenotes
- https://lecturenotes.in/notes/14662-note-for-principles-of-programming-languages-popl-by-jntu-heroes?reading=true