Indra Ganesan
COLLEGE OF ENGINEERING

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai
Accredited by NAAC with 'B+' Grade, 2(f) & 12B Status Institution by UGC
IG Valley, Madurai Main Road, Manikandam, Tiruchirappalli - 620012

NAAC DOCUMENTS

QUALITY INDICATOR FRAME WORK

CRITERION – 1

CURRICULAR ASPECTS

SUBMITTED BY

IQAC

INTERNAL QUALITY ASSURANCE CELL

INDRA GANESAN COLLEGE OF ENGINEERING

Criteria 1	Curricular Aspects	100

1.1 Curricular Planning and Implementation (20)

1.1.1 The Institution ensures effective curriculum planning and delivery through a well-planned and documented process including Academic calendar and conduct of continuous internal Assessment

Table of Content

INDRA GANESAN COLLEGE OF ENGINEERING

IG Valley, Manikandam, Tiruchirappalli, Tamil Nadu 620012 , India
(Approved by AICTE, NewDelhi, Affiliated to AnnaUniversity, Chennai-25)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PREFACE OF THE COURSE FILE

Batch	: **2018-2019**
Academic Year	: **2018-2019/ODD**
Program	: **M.E. COMPUTER SCIENCE AND ENGINEERING**
Year & Semester	: **1ST Year/1ST Semester/'A'Section**
Course Code	: **CP5151**
NBA Course Code	: **C102**
Name of the Course	: **ADVANCED DATA STRUCTURES AND ALGORITHMS**
Faculty in-charge	: **Ms.J.Jenifer/Asst.Prof/CSE**

Signature of the Faculty in-charge

HoD/ CSE

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

INDRA GANESAN COLLEGE OF ENGINEERING

(Approved by AICTE, New Delhi and affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

REVIEW OF COURSE FILE

(To be pasted on the inner side of the file-backside).(#-State Yes/No.)

S.N	Details Date:	R-I-*	R-II-*&	R-III-*&	R-IV-*&$	R-V-*&$@
1.	Preface of the course file	Y				
2.	Vision,Mission,PEOs,POs,PSOs,Blooms taxonomy	Y				
3.	Subject handlers of yesteryears	Y				
4.	Timetable/Workloadofthestaff–Distributionof teachingload–Rolesand Responsibilities	Y				
5.	Syllabussignedbystaff&HoD	Y				
6.	LectureScheduleSignedbystaff&HoD	Y				
7.	CourseCommitteemeetingcircularandminutes	Y				
8.	IdentificationofCurriculargapandContent Beyondthesyllabus	Y				
9.	Self-studytopics	Y				
10.	PreviousAUQuestionpapers	Y				
11.	UnitwiseQ&AandObjectivetypequestions	Y				
12.	Unitwisecoursematerial	Y				
13.	Assignmentquestionpaperwithsampleanswersheetsan dmarkentry		Y	Y	Y	
14.	Tutorialquestionpaperwithkeyandmarkentry		Y	Y	Y	
15.	Classtest/IAtestQPaperwithKey,sample answerpapersandmarkentry		Y	Y	Y	
16.	IATest-resultanalysis-CAP-evidence-rootcause analysis.		Y	Y	Y	
17.	Retest–Qpaper-Attendance-marks		Y	Y	Y	
18.	AUWebportalentrysheet		Y	Y	Y	
19.	Verypoorperformanceinfirsttwotests-action taken.-communicationtoparents-evidence		Y	Y	Y	
20.	Absencefortwotests-actiontaken-communication toparents-evidence.		Y	Y	Y	
21.	Indisciplineofstudentreported,ifany			Y	Y	
22.	Specialclass/coachingclass/remedial class/attendance-CAP			Y	Y	
23.	ConductofSeminar,Quizzes-proof					
24.	Content beyond the syllabus-proof		Y	Y	Y	
25.	Student feedback on faculty					Y
26.	Course end survey					Y
27.	Internal Assessment sheet					Y
28.	AU question paper with students feedback					Y
29.	Discrepancyofthequestionpaperandcorrespondence,if any					Y
30.	AU result analysis-Detail sofarrearstudents.					Y
31.	AU grade sheet					Y
32.	CO–PO&PSOattainmentsheet					Y
	Signature of Course handling faculty					
	Signature of HoD	D. Gobi	D. Gobi	D. Gobi	D. Gobi	D. Gobi

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

INDRA GANESAN COLLEGE OF ENGINEERING

IG Valley,Manikandam,Tiruchirappalli,TamilNadu–620 012 , India
(Approved by AICTE,NewDelhi,Affiliated to Anna University,
Chennai-25)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Faculty Time Table

Ms. J. Jenifer AP/CSE

Day Order	1	2	3	4	5	6	7	8
I			ADA					
II								
III				ADA				
IV							ADA	
V	ADA							

S. Code	Title	Year/Branch	Hours
CP5151	ADVANCEDDATASTRUCTURESAND ALGORITHMS	I/CSEA	4 Hours
TOTAL- 4hours			

Signature of the Faculty

Hod/CSE

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

CP5151 ADVANCED DATA STRUCTURES AND ALGORITHMS **LTPC**

3003

OBJECTIVES:

- To understand the usage of algorithms in computing.
- To learn and use hierarchical data structures and its operations.
- To learn the usage of graphs andits applications..
- To select and design data structures and algorithms that is appropriate for problems.
- To study about NP Completeness of problems.

UNIT I ROLE OF ALGORITHMS INCOMPUTING **12**

Algorithms–AlgorithmsasaTechnology-InsertionSort–Analyzing Algorithms–Designing Algorithms- Growth of Functions: Asymptotic Notation – Standard Notations and Common Functions- Recurrences: The Substitution Method – The Recursion-Tree Method.

UNIT II HIERARCHICAL DATA STRUCTURES **12**

Binary Search Trees: Basics – Querying a Binary search tree – Insertion and Deletion- Red-Black trees: Properties of Red-Black Trees – Rotations – Insertion – Deletion -B-Trees: Definition of Btrees– BasicoperationsonB-Trees– Deleting akeyfromaB-Tree- FibonacciHeaps: structure – Mergeable-heapoperations- Decreasingakeyanddeletinganode-Boundingthemaximumdegree.

UNIT III GRAPHS **12**

Elementary Graph Algorithms: Representations of Graphs – Breadth-First Search – Depth-First Search–TopologicalSort –StronglyConnectedComponents- MinimumSpanningTrees:Growing a Minimum Spanning Tree – Kruskal and Prim- Single-Source Shortest Paths: The Bellman-Ford algorithm – Single-Source Shortest paths in Directed Acyclic Graphs – Dijkstra's Algorithm; All-Pairs Shortest Paths: Shortest Paths and Matrix Multiplication – The FloydWarshall Algorithm;

UNIT IV ALGORITHM DESIGN TECHNIQUES **12**

Dynamic Programming: Matrix-Chain Multiplication – Elements of Dynamic Programming – Longest Common Subsequence- Greedy Algorithms: An Activity-Selection Problem – Elements of the Greedy Strategy- Huffman Codes.

UNIT V NP COMPLETE AND NP HARD **12**

NP-Completeness:PolynomialTime–Polynomial-TimeVerification–NP-Completenessand Reducability – NP-Completeness Proofs – NP-Complete Problems.

Total:60 PERIODS

OUTCOMES:

At the end of the course, the students should be able to:

- Design data structures and algorithms to solve computing problems.
- Design algorithms using graph structure and various string matching algorithms to solve real-life problems.
- Apply suitable design strategy for problem solving.

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

TEXTBOOKS:

1. Alfred .Aho,JohnE.Hopcroft,JeffreyD.Ullman,—DataStructuresand Algorithms‖, Pearson Education, Reprint 2006.
2. Robert SedgewickandKevinWayne,—ALGORITHMS‖,FourthEdition,Pearson Education.
3. S.Sridhar,‖DesignandAnalysisofAlgorithms‖,FirstEdition,OxfordUniversityPress. 2014.
4. ThomasH.Cormen,CharlesE.Leiserson,RonaldL.Rivest,CliffordStein,—Introduction to Algorithms‖, Third Edition, Prentice-Hall, 2011.

REFERENCES:

1. AlfredV.Aho,JohnE.Hopcroft,JeffreyD.Ullman,—DataStructuresand Algorithms‖, Pearson Education, Reprint 2006.
2. RobertSedgewickandKevinWayne,—ALGORITHMS‖,FourthEdition,PearsonEducation.
3. S.Sridhar,‖Designand AnalysisofAlgorithms‖,FirstEdition,OxfordUniversityPress.2014.
4. ThomasH.Cormen,CharlesE.Leiserson,RonaldL.Rivest,CliffordStein,—Introductionto Algorithms‖, Third Edition, Prentice-Hall, 2011.

Signature of the Faculty Incharge

HOD/CSE

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

INDRA GANESAN COLLEGE OF ENGINEERING

IG Valley,Manikandam,Tiruchirappalli,TamilNadu–620012 India
Approved by AICTE, NewDelhi ,Affiliated to AnnaUniversity,Chennai-25)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Lecture Schedule

Degree/Program:**M.E/CSE**

CourseCode: **CP5151**

CourseName:**ADVANCED DATA STRUCTURES AND ALGORITHMS**

Duration: **2018-2019** Semester:**I** Section:**A** Faculty:Ms. J. JeniferAP/CSE

AIM:

To expose the student's principles of logical data structures and its algorithms

OBJECTIVES:

- To impart knowledge on
- To understand the usage of algorithms in computing.
- To learn and use hierarchical data structures and its operations
- To learn the usage of graphs and its applications.
- To select and design data structures and algorithms that is appropriate for problems.

PREREQUISITES: Graphical Structure theory, Problem Solving algorithms.

AIM:

To expose the student's principles of logical data structures and its algorithms

COURSE OUTCOMES:

After the course, the student should be able to:

CO	Course Outcomes	POs	PSOs
C102.1	Designdatastructuresandalgorithmstosolvecomputing problems.	1,2,3,4,5,6,7,8,9,10,11,12	1,2
C102.2	Designalgorithmsusinggraphstructure.	1,2,3,4,5,6,7,8,9,10,11,12	1,2
C102.3	Designvariousstringmatchingalgorithmstosolvereal-life problems.	1,2,3,4,5,6,7,8,9,10,11,12	1,2
C102.4	Applysuitabledesignstrategyforproblemsolving	1,2,3,4,5,6,7,8,9,10,11,12	1,2
C102.5	Developcompletenessofalgorithmtechniques&strategies.	1,2,3,4,5,6,7,8,9,10,11,12	1,2

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

S.No	Date	Topics to be Covered	Book
UNIT–I ROLE OF ALGORITHMS IN COMPUTING periods:12			
1	6.7.18	Algorithms	T1/BB
2	7.7.18	AlgorithmsasaTechnology	R2/BB
3	7.7.18	InsertionSort	T1/BB
4	8.7.18	AnalyzingAlgorithms	T3/BB
5	9.7.18	DesigningAlgorithms	R3/BB
6	10.7.18	Growthof Functions	T2/BB
7	15.7.18	AsymptoticNotation	T1/BB
8	16.7.18	StandardNotations	T1/BB
9	17.7.18	CommonFunctions	T1/BB
10	21.7.18	RecurrencesandSubstitutionMethod	
11	21.7.18	TheRecursionandTreeMethod	
UNIT II- HIERARCHICAL DATA STRUCTURES periods:12			
12	28.7.18	BinarySearchTrees.	T1/BB
13	01.8.18	BasicsandQueryingaBinarysearchtree	R2,T1/BB
14	03.8.18	InsertionandDeletion	R2,T1/BB
15	16.8.18	Red-BlacktreesandPropertiesofRed-BlackTrees:	T1/BB
16	20.8.18	Rotations,Insertion,Deletion	R3/BB
17	21.8.18	B-Trees:DefinitionofBtrees	T1/BB
18	24.8.18	BasicoperationsonB-TreesandDeletingakeyfromaB-Tree	T1/BB
19	27.8.18	FibonacciHeaps:structureandMergeable	R1/BB
20	27.8.18	heapoperations	T1/BB
21	28.8.18	Decreasingakeyanddeletinganode	
22	28.8.18	Boundingthemaximumdegree	
UNIT III- GRAPHS Periods:12			
23	30.8.18	ElementaryGraphAlgorithmsandRepresentations ofGraphs	T1/BB
24	31.8.18	BreadthFirst SearchandDepthFirstSearch	T1/BB T1/BB
25	3.9.18	TopologicalSort	R1/BB
26	5.9.18	StronglyConnectedComponents	T2/BB
27	6.9.18	MinimumSpanningTrees	R1/BB
28	10.9.18	GrowingaMinimumSpanningTree	T3/BB
39	12.9.18	KruskalandPrim- Single	T3/BB
30	24.9.18	SourceShortestPaths:TheBellmanandFordalgorithm	
31	24.9.18	SingleandSourceShortest pathsinDirectedAcyclicGraphsandDijkstra's Algorithm	
32	26.9.18	AllandPairsShortestPaths:ShortestPathsandMatrixMultiplicationand TheFloydWarshall Algorithm	
UNIT IV-ALGORITHM DESIGN TECHNIQUES Target Periods:12			
33	26.9.18	DynamicProgramming	T1/BB
34	28.9.18	Matrix	T1/BB
35	28.9.18	ChainMultiplication	R2/BB
36	2.10.18	ElementsofDynamicProgramming	T1/BB
37	3.10.18	LongestCommonSubsequence	T3/BB
38	4.10.18	GreedyAlgorithms	R3/BB
39	8.10.18	AnActivityandSelectionProblem	R2/BB
40	10.10.18	SelectionProblem	T1/BB
41	10.10.18	ElementsoftheGreedyStrategy	

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

42	12.10.18	HuffmanCodes	
UNIT V – NP COMPLETE AND NP HARD		**Target Periods:12**	
43	12.10.18	NP-Completeness	T1/BB
44	15.10.18	PolynomialTime	T2/BB
45	15.10.18	Polynomial	R1/BB
46	16.10.18	TimeVerification	T3/BB
47	16.10.18	NP-CompletenessandReducability	R3/BB
48	17.10.18	NP-CompletenessProofs	T1/BB
49	17.10.18	NP-Completenessidentcation	R2/BB
50	18.10.18	NP-Completenessrootcauses	R1/BB
51	18.10.18	NP-CompletenessProblems	
Content Beyond the Syllabus			
54	27.10.18	REHASHING	Material

TEXTBOOKS:

1. AlfredV.Aho,JohnE.Hopcroft,JeffreyD.Ullman,—DataStructuresand Algorithmsl, Pearson Education, Reprint 2006.
2. RobertSedgewickandKevinWayne,—ALGORITHMSl,FourthEdition,Pearson Education.
3. S.Sridhar,lDesignandAnalysisofAlgorithmsl,FirstEdition,OxfordUniversityPress.2014
4. ThomasH.Cormen,CharlesE.Leiserson,RonaldL.Rivest,CliffordStein,—Introduction to Algorithmsl, Third Edition, Prentice-Hall, 2011.

REFERENCES:

1. AlfredV.Aho,JohnE.Hopcroft,JeffreyD.Ullman,—DataStructuresand Algorithmsl, Pearson Education, Reprint 2006.
2. RobertSedgewickandKevinWayne,—ALGORITHMSl,FourthEdition,Pearson Education.
3. S.Sridhar,lDesignandAnalysisofAlgorithmsl,FirstEdition,OxfordUniversityPress.2014
4. ThomasH.Cormen,CharlesE.Leiserson,RonaldL.Rivest,CliffordStein,—Introduction to Algorithmsl, Third Edition, Prentice-Hall, 2011

Signature of the Faculty in-charge

HoD / CSE

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

INDRA GANESAN COLLEGE OF ENGINEERING

IG Valley, Manikandam, Tiruchirappalli, Tamil Nadu – 620012, India
(Approved by AICTE, New Delhi, Affiliated toAnnaUniversity,Chennai-25)

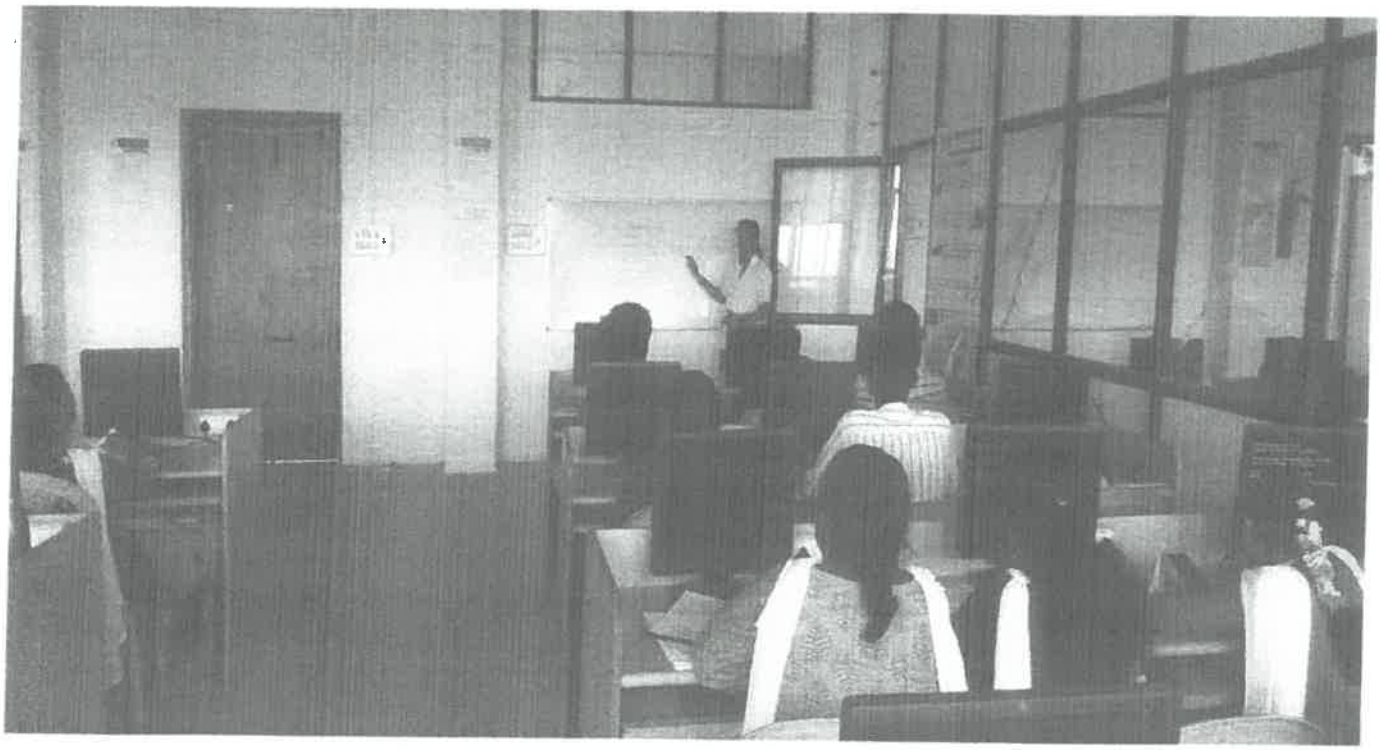
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CBS-PROOF

ACADEMICYEAR: 2018-2019(ODD) **SEM:01** **REGULATION:2017**

PROGRAM: M.E. CSE **Name of the Faculty: Ms. J. Jenifer**

TOPIC: REHASHING SPACE COMPLEXITY



Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

INDRA GANESAN COLLEGE OF ENGINEERING

IG Valley, Manikandam, Tiruchirappalli, Tamil Nadu – 620 012, India
(Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai-25)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Assignment Answer Sheet

Name of the Student: Gwendolyn Rosetta . G

AU Register Number: 811218405002

Assignment – 01				Date of Issue:	9.10.2018	Marks	10
Course code	CP5151	Course Title	ADA				
Year	2018	Semester/Section	I / A	Date of Submission:		5.11.2018	

Q.No	Questions	CO
1	Analyse Quick Sort, Merge Sort algorithm	2
2	Explain the use of DAC technique for binary search method? Solve its recurrence relation $T(n) = T(n/2) + 1$	3

Mark Allocation

Rubrics	Marks Allocated	Marks obtained
Content Quality	6	4
Presentation Quality	2	1
Timely submission	2	2
Total marks	10	7

Name and Signature of the Faculty Incharge

HoD/CSE

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

INDRA GANESAN COLLEGE OF ENGINEERING

IG Valley, Manikandam, Tiruchirappalli, Tamil Nadu – 620 012, India
(Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai-25)

IQAC Academic Audit Form

ACADEMIC YEAR: 2018-2019 ODD SEMESTER

Name of Department : M.E. CSE | Year / Sem / Sec : I/I | No. of Students Registered : 5

Details of Examination : IA Test -1

S.No.	Course Code	List of Reg.No Varified	Course Log Book Varified (Y / N)	Course File Verified (Y / N)	No of students Attended	No of Absentees	No of Failures	Pass %	Remarks
1.	MA5160	811218405001, 811218405004	Y	Y	4	1	-	98%	—
2.	CP5151	811218405001, 811218405004	Y	Y	4	1	-	98%	—
3.	CP5152	811218405001, 811218405004	Y	Y	4	1	-	98%	—
4.	CP5153	811218405001, 811218405004	Y	Y	4	1	-	98%	—
5.	CP5154	811218405001, 811218405004	Y	Y	4	-	-	100%	—
6.	CP5191	811218405001, 811218405004	Y	Y	4	-	-	100%	—

Verified by

External Member Name and Signature:

Internal Member Name and Signature:

Overall Remarks:

HoD/ CSE

IQAC Co-ordinator

Principal

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

Register Number:											

INDRA GANESAN COLLEGE OF ENGINEERING

IGValley,Manikandam,Tiruchirappalli,TamilNadu–620012,India

(Approved by AICTE, New Delhi and affiliated to Anna University, Chennai)

Internal Assessment Exam-I			Date/Session		Marks	50
Course code	CP5151	Course Title	Advanced Data Structures and algorithms			
Regulation	2017	Duration	90 minutes	AcademicYear	2018-2019	
Year	2018/I	Semester	I	Department	M.E.CSE	

COURSE OUTCOMES

CO1:	Design data structures and algorithms to solve computing problems
CO2:	Design algorithms using graph structure
CO3:	Design various string matching algorithms to solve real-life problems
CO4:	Apply suitable design strategy for problem solving
CO5:	Algorithm design techniques solved
CO6:	NP–Completeness of problem solving

Q.No.	Question	CO	BTS
	PARTA		
	(AnsweralltheQuestions10x2=20Marks)		
1	What is called as a Data Structure?	1	1
2	Differentiate between data type and data structures.	2	1
3	What is a postfix expression?.	2	1
4	What is a queue Data Structure?	2	2
5	Write an algorithm to count the nodes in a circular queue.	1	1
6	Define the terms: Infix, postfix and prefix.	2	2
7	Define Linked list and give its applications.	2	2
8	Define Full binary tree(or) Complete binary tree.	2	2
9	Define expression tree.	2	2
10	Traverse the given tree using Inorder, Preorder and Postorder traversals.	3	2
	PARTB		
	(AnswerallthQuestions2x10=20Marks)		
11a	Write the algorithm for performing operations in a stack. Trace your algorithm with suitable example.	2	2
	OR		
11b	Explain the algorithm for implementing Singly Linked list.	2	2
12a	Explain creation, insertion and deletion of doubly linked list with example.	2	2
	OR		
12b	Construct a dequeue data structure in which the following operations to be implemented.	2	2
	PARTC		
	(AnswerallthQuestions1x10=10Marks)		
13a	What is Binary search tree? Write an algorithm to add a node into a binary search tree.	2	2
	OR		
13b	Construct a minimum spanning tree using Kruskal's algorithm with your own example	2	3

Course Faculty

(Name/Sign/Date)

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

HoD

(Name/Sign/Date)

Register Number:											

INDRA GANESAN COLLEGE OF ENGINEERING

IGValley,Manikandam,Tiruchirappalli,TamilNadu–620012,India

(Approved by AICTE, New Delhi and affiliated to Anna University, Chennai)

Internal Assessment Exam-I			Date/Session		Marks	50
Course code	CP5151	Course Title	Advanced Data Structures and algorithms			
Regulation	2017	Duration	90 minutes	AcademicYear	2018-2019	
Year	2018/I	Semester	I	Department	M.E.CSE	

COURSE OUTCOMES

CO1:	Design data structures and algorithms to solve computing problems
CO2:	Design algorithms using graph structure
CO3:	Design various string matching algorithms to solve real-life problems
CO4:	Apply suitable design strategy for problem solving
CO5:	Algorithm design techniques solved
CO6:	NP–Completeness of problem solving

Q.No.	Questions and Answer keys	CO	BTS
	PART A (Answer all the Questions 10x2=20Marks)		
1	**What is called as a Data Structure?** A data structure is a way of organizing and storing data to perform operations efficiently. A data structure is a specialized format or arrangement of data that allows efficient storage, organization, and retrieval of information in a computer system.	1	1
2	**Differentiate between data type and data structures.**	2	1

Aspect	Data Type	Data Structure
Definition	Describes the type of data values	Organizes and stores data efficiently
Focus	Specifies the nature of individual	Focuses on the arrangement and man
Examples	int, float, char, boolean, etc.	Array, Linked List, Stack, Queue, etc.
Usage	Used for declaring variables	Used for organizing and manipulating
Level of Abstraction	Low-level	High-level
Relationship	Building blocks for data structures	Utilizes data types to define compone

3	**What is a postfix expression?** A postfix expression, also known as Reverse Polish Notation (RPN), is a mathematical notation in which operators follow their operands. In other words, instead of using traditional infix notation where operators are placed between operands (e.g., 2 + 3), postfix notation places operators after operands (e.g., 2 3 +). This eliminates the need for parentheses and clarifies the order of operations. Postfix expressions are often used in computer science, especially in stack-based algorithms and calculators.	2	1
4	**What is a queue Data Structure?** A queue is a data structure that follows the First In, First Out (FIFO) principle. In a queue, elements are added at the rear (enqueue) and removed from the front (dequeue). This ensures that the element that has been in the queue the longest is the first one to be processed or removed. It operates like a real-world queue or line, where the first person to join is the first to be served. Queues are commonly used in various computer science applications, such as task scheduling, breadth-first search algorithms, and managing resources in a sequential manner.	2	2
5	**Write an algorithm to count the nodes in a circular queue.** Certainly! Here's a simple algorithm in pseudocode to count the nodes in a circular queue: Function CountNodes(circularQueue): if circularQueue is empty:	1	1

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

	current = circularQueue.front.next // Start from the next node of the front while current is not equal to circularQueue.front: count = count + 1 current = current.next return count. This algorithm assumes that the circular queue has a structure where each node has a 'next' pointer that points to the next node in the circular sequence, and the last node's 'next' pointer points back to the front of the queue. The 'front' points to the first node in the queue. The algorithm traverses the circular queue, counting nodes until it reaches the front again. If the queue is empty, it returns 0.		
6	**Define the terms: Infix, postfix and prefix.** In the context of expressions: 1. **Infix:** Infix notation is the standard way of writing mathematical expressions where operators are placed between operands. For example, in the infix expression "a + b", the operator '+' is placed between the operands 'a' and 'b'. 2. **Postfix (or Reverse Polish Notation - RPN):** Postfix notation is a way of writing mathematical expressions in which the operators come after their operands. For example, in postfix notation, "a + b" is written as "a b +". Evaluation of postfix expressions is often done using a stack data structure. 3. **Prefix (or Polish Notation):** Prefix notation is a way of writing mathematical expressions in which the operators precede their operands. For example, in prefix notation, "a + b" is written as "+ a b". Like postfix notation, the evaluation of prefix expressions is commonly done using a stack.	2	2
7	**Define Linked list and give its applications.** A linked list is a linear data structure where elements are stored in nodes, and each node points to the next node in the sequence, forming a chain-like structure. Applications of Linked Lists: 1. **Dynamic Memory Allocation:** Linked lists allow dynamic memory allocation, making them useful for managing memory efficiently. 2. **Implementation of Data Structures:** Linked lists are used to implement various data structures like stacks, queues, and symbol tables. 3. **Undo Functionality:** Linked lists can be employed to implement undo functionality in applications where a history of operations needs to be maintained. 4. **File Systems:** Linked lists are used in file systems to keep track of the blocks of a file scattered across the storage. 5. **Representing Sparse Data:** Linked lists are suitable for representing sparse data structures where most elements are empty, saving memory. 6. **Polynomial Manipulation:** Linked lists are used to represent and manipulate polynomials efficiently. 7. **Graph Algorithms:** Linked lists can be utilized to represent adjacency lists in graph-related algorithms.	2	2
8	**Define Full binary tree(or) Complete binary tree.** A full binary tree is a binary tree in which every node has either 0 or 2 children. In other words, every level of the tree is completely filled, except possibly for the last level, which is filled from left to right. A complete binary tree is a special case of a binary tree where all levels are filled except possibly the last, and all nodes are as left as possible.	2	2
9	**Define expression tree.** An expression tree is a binary tree representation of an arithmetic expression, where the leaves are operands and the internal nodes are operators. The tree's structure reflects the hierarchy and order of operations in the original expression, making it a useful data structure for evaluating and manipulating mathematical expressions.	2	2
10	**Traverse the given tree using Inorder, Preorder and Postorder traversals.** Here are brief descriptions of the three tree traversal methods: 1. **Inorder Traversal:** • Traverse the left subtree in inorder. • Visit the root node. • Traverse the right subtree in inorder.	3	2

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

```
    new_node = Node(value)
    if head is None:
        head = tail = new_node
    else:
        new_node.prev = tail
        tail.next = new_node
        tail = new_node

# Deletion from the End
def delete_at_end():
    global head, tail
    if head is None:
        print("Underflow: List is empty.")
    elif head == tail:
        head = tail = None
    else:
        tail = tail.prev
        tail.next = None

# Display Doubly Linked List
def display():
    current = head
    while current:
        print(current.data, end=" <-> ")
        current = current.next
    print("NULL")

# Example Usage
create_doubly_linked_list()

insert_at_end(1)
insert_at_end(2)
insert_at_end(3)

print("Doubly Linked List after insertion:")
display()

delete_at_end()

print("Doubly Linked List after deletion:")
display().
```

This example demonstrates the creation of a doubly linked list, insertion at the end, and deletion from the end. The display function shows the state of the doubly linked list after each operation.

OR

12b **Construct a dequeue data structure in which the following operations to be implemented.** 2 2

A deque (double-ended queue) is a data structure that allows elements to be added or removed from both ends. Here's an example implementation in Python using a doubly linked list:

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
        self.prev = None

class Deque:
    def __init__(self):
```

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

	new_node.next = head head = new_node # Insert elements at the end temp = head while temp.next is not None: temp = temp.next new_node = Node(5) temp.next = new_node # Delete from the beginning head = head.next # Traversal temp = head while temp is not None: print(temp.data, end=" ") temp = temp.next. This example demonstrates the basic operations of inserting at the beginning and end, deleting from the beginning, and traversing a singly linked list.		
12a	**Explain creation, insertion and deletion of doubly linked list with example.** Let's go through the creation, insertion, and deletion of nodes in a doubly linked list along with an example. 1. **Creation of Doubly Linked List:** • Initialize pointers head and tail to NULL. • For each element to be inserted, create a new node. • If the list is empty, set both head and tail to the new node. • Otherwise, update pointers to maintain the doubly linked structure. 2. **Insertion in Doubly Linked List:** • Insertion can occur at the beginning, end, or at a specific position. • For simplicity, let's consider inserting at the end. • Create a new node. • If the list is empty, set both head and tail to the new node. • Otherwise, update pointers accordingly. 3. **Deletion in Doubly Linked List:** • Deletion can occur at the beginning, end, or at a specific position. • For simplicity, let's consider deleting at the end. • If the list is empty, indicate underflow. • Otherwise, update pointers accordingly. **Here's an example in Python:** class Node: def __init__(self, data): self.data = data self.next = None self.prev = None # Creation of Doubly Linked List def create_doubly_linked_list(): global head, tail head = tail = None # Insertion at the End def insert_at_end(value): global head, tail	2	2

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

11b	**Explain the algorithm for implementing Singly Linked list.** Certainly! Implementing a singly linked list involves creating nodes and linking them in a forward direction. Here's a basic algorithm for the operations on a singly linked list: **Algorithm for Singly Linked List Operations:** 1. **Define Node structure:** - Each node contains a data element and a reference (or pointer) to the next node. 2. **Initialization:** - Set the head pointer to null initially, indicating an empty list. 3. **Insertion at the beginning (InsertFront):** - Create a new node with the given data. - Set the next pointer of the new node to the current head. - Update the head pointer to the new node. 4. **Insertion at the end (InsertEnd):** - Create a new node with the given data. - Traverse the list until the last node. - Set the next pointer of the last node to the new node. 5. **Deletion from the beginning (DeleteFront):** - If the list is empty, indicate underflow. - Otherwise, update the head pointer to the next node. 6. **Traversal:** - Start from the head. - While the current node is not null, process the data and move to the next node. 7. **Search (optional):** - Start from the head. - While the current node is not null, check if the data matches the target. - Return true if found, false otherwise. 8. **Deletion by value (optional):** - Start from the head. - Keep track of the current and previous nodes. - If the target value is found, update the next pointer of the previous node to skip the current node. Note: Always handle edge cases such as an empty list, and manage pointers carefully to avoid memory leaks. Example: ```python class Node: def __init__(self, data): self.data = data self.next = None # Initialize an empty linked list head = None # Insert elements at the beginning new_node = Node(3) new_node.next = head head = new_node new_node = Node(2)	2	2

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

	2. **Preorder Traversal:** • Visit the root node. • Traverse the left subtree in preorder. • Traverse the right subtree in preorder. 3. **Postorder Traversal:** • Traverse the left subtree in postorder. • Traverse the right subtree in postorder. • Visit the root node. These traversal methods can be applied to any tree, and the order in which nodes are visited defines the respective traversal. The exact sequence of nodes visited will depend on the specific structure of the tree.		
	PARTB **(AnswerallthQuestions2x10=20Marks)**		
11a	**Write the algorithm for performing operations in a stack. Trace your algorithm with suitable Example.** Certainly! Let's consider a simple algorithm for performing operations (push and pop) in a stack and trace it with an example: **Algorithm for Stack Operations:** 1. Initialize an empty stack. 2. Push (Insert): - Check if the stack is full (overflow). - If not full, insert the element onto the top of the stack. 3. Pop (Delete): - Check if the stack is empty (underflow). - If not empty, remove the element from the top of the stack. Example: Let's perform operations on a stack with a maximum capacity of 5: 1. Initialize an empty stack: `Stack = []` 2. Push elements onto the stack: - Push 1: `Stack = [1]` - Push 2: `Stack = [1, 2]` - Push 3: `Stack = [1, 2, 3]` 3. Pop elements from the stack: - Pop: `Stack = [1, 2]` - Pop: `Stack = [1]` 4. Push more elements: - Push 4: `Stack = [1, 4]` - Push 5: `Stack = [1, 4, 5]` 5. Pop again: - Pop: `Stack = [1, 4]` This algorithm ensures that the stack is not accessed or modified in case of underflow or overflow, respectively. It follows the Last-In-First-Out (LIFO) principle, where the last element pushed is the first one to be popped.	2	2
	OR		

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012

	subtree. The algorithm modifies the tree and returns the root of the modified BST.		
	OR		
13b	**Construct a minimum spanning tree using Kruskal's algorithm with your own example.**	2	3

Kruskal's Algorithm for Minimum Spanning Tree (MST):

Kruskal's algorithm is a greedy algorithm that finds a minimum spanning tree for a connected, undirected graph. It works by sorting the edges in ascending order of their weights and adding edges to the MST while avoiding the formation of cycles.

Let's consider a simple graph and apply Kruskal's algorithm step by step:

Graph:

```
A - 2 - B
|     |
1     3
|     |
C - 4 – D
```

Steps:

Sort Edges by Weight:

- (A, B): 2
- (C, D): 4
- (A, C): 1
- (B, D): 3

Initialize Empty MST:

- MST = {}

Add Edges to MST:

- Add (A, C): 1
  - MST = {(A, C)}
- Add (A, B): 2
  - MST = {(A, C), (A, B)}
- Add (B, D): 3
  - MST = {(A, C), (A, B), (B, D)}

The final Minimum Spanning Tree (MST) contains the edges {(A, C), (A, B), (B, D)} with a total weight of 6.

Note: Kruskal's algorithm can handle graphs with weighted edges, and in case of ties in edge weights, the choice of which edge to add first is arbitrary. The key is to avoid creating cycles in the process.

Course Faculty

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

HoD

```
        self.front = None
        self.rear = None

    def is_empty(self):
        return self.front is None

    def add_front(self, data):
        new_node = Node(data)
        if self.is_empty():
            self.front = self.rear = new_node
        else:
            new_node.next = self.front
            self.front.prev = new_node
            self.front = new_node

    def add_rear(self, data):
        new_node = Node(data)
        if self.is_empty():
            self.front = self.rear = new_node
        else:
            new_node.prev = self.rear
            self.rear.next = new_node
            self.rear = new_node

    def remove_front(self):
        if self.is_empty():
            return None
        data = self.front.data
        if self.front == self.rear:
            self.front = self.rear = None
        else:
            self.front = self.front.next
            self.front.prev = None
        return data

    def remove_rear(self):
        if self.is_empty():
            return None
        data = self.rear.data
        if self.front == self.rear:
            self.front = self.rear = None
        else:
            self.rear = self.rear.prev
            self.rear.next = None
        return data

    def display(self):
        current = self.front
        while current:
            print(current.data, end=" ")
            current = current.next
        print()

# Example Usage:
deque = Deque()

deque.add_front(1)
deque.add_front(2)
deque.add_rear(3)
deque.add_rear(4)

deque.display()  # Output: 2 1 3 4
```

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

```
print("Removed front:", deque.remove_front())  # Output: 2
print("Removed rear:", deque.remove_rear())    # Output: 4

deque.display()  # Output: 1 3
```

In this example, **`add front`** and **`remove front`** operations manipulate the front of the deque, while **`add rear`** and **`remove rear`** operations manipulate the rear. The doubly linked list structure allows easy addition and removal of elements from both ends of the deque.

PART C
(Answer all the Questions 1x10=10 Marks)

13a **What is Binary search tree? Write an algorithm to add a node into a binary search tree.** — 2 — 2

A Binary Search Tree (BST) is a binary tree data structure in which each node has at most two children, referred to as the left child and the right child. For every node, all elements in its left subtree are less than the node, and all elements in its right subtree are greater than the node.

Algorithm to Add a Node to a Binary Search Tree:

Algorithm: InsertNode
Input: Root of the BST, Value to be inserted

1. If the root is null, create a new node with the given value and set it as the root.
2. Otherwise:
   a. If the value to be inserted is less than the root's value:
      - Recursively call InsertNode on the left subtree.
   b. If the value to be inserted is greater than the root's value:
      - Recursively call InsertNode on the right subtree.
3. Return the modified root.

Example in Python:

```
class TreeNode:
    def __init__(self, key):
        self.left = None
        self.right = None
        self.val = key

def insert_node(root, key):
    if root is None:
        return TreeNode(key)
    else:
        if key < root.val:
            root.left = insert_node(root.left, key)
        elif key > root.val:
            root.right = insert_node(root.right, key)
    return root

# Example usage:
root = None
keys = [50, 30, 20, 40, 70, 60, 80]

for key in keys:
    root = insert_node(root, key).
```

In this example, the **`insert node`** algorithm recursively traverses the BST to find the appropriate position for the new node based on the comparison of values. If the value is less than the current node, it goes to the left subtree; if it is greater, it goes to the right

Dr. G. Balakrishnan, M.E., Ph.D.
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

INDRA GANESAN COLLEGE OF ENGINEERING

IG Valley,Manikandam,Tiruchirappalli,TamilNadu–622012,India
(Approved by AICTE,NewDelhi and affiliated to AnnaUniversity, Chennai)

Internal Assessment Test Answer Book

Name	Gwendolyn Rosetta . G			**Year/Semester/Section**	I/I
Batch No.	8112184050002	**Date/Session**	8.9.2018	**Department**	M.E.CSE
Course code	CP5151	**Course Title**	Advance Data Structures & Algorithms		
Internal Assessment Test		**IAT1** ☐	**IAT2** ☐	**IAT3** |	**Model** |
Name and Signature of the Invigilator with date					

Instruction to the Student: Put tick mark to the question attended in the column against question.

PartA			PartB/PartC						
Q.No.	✓	Marks	Q.NO.	✓	a Marks	✓	b Marks	TotalMarks
1		2	11		10			10
2		2	12		6			6
3		1	13				6	6
4		0	14					
5		2	15					
6		2	16					
7		2					Total	22
8		2						
9		2						
10		2						
Total		18	40 Grand Total				Name and Signature of the Examiner with date	

To be filled by the examiner							
Course Outcomes	1	2	3	4	5	6	Total
Marks allotted							
Marks Obtained							

IQACAudit-Remarks

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

NameandSignatureof the IQAC member

INDRA GANESAN COLLEGE OF ENGINEERING

IG VALLEY, MANIDANDAM, TIRUCHIRAPPALLI – 620 012

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ACADEMIC YEAR 2022 – 2023 (ODD SEMESTER)

STUDENTS MARK STATEMENT- CO BASED
INTERNAL ASSESSMENT TEST-1

SUBJECT CODE &TITLE: CP5151& ADVANCED DATASTRUCTURES AND ALGORITHMS

YEAR/SEM: I/I **MONTH & YEAR:**

S.NO	REG NO	STUDENT NAME	COX (32)	COX (18)	TOTAL (50)	TOTAL (100)
1.	81218405001	Aswini. M	30	15	45	90
2.	81218405002	Gowenddeyn Rosetta.G	25	10	35	70
3.	81218405003	Harish. V	20	9	29	58
4.	81218405004	Nirmala. N	24	12	36	72

MARKS RANGE:

<20	20-30	31-40	41-50	51-60	61-70	71-80	81-90	91-100
				1	1	1	1	

Total No.of Candidates Present	4
Total No.of Candidates Absent	0
Total No.of Students Pass	4
Total No. of Students Fail	0
Percentage of Pass	100 %

Dr. G. Balakrishnan, M.E., Ph.D.,
Principal
Indra Ganesan College of Engineering
IG Valley, Madurai Main Road
Manikandam, Trichy-620 012.

Signature of the Faculty In-charge

HOD/CSE