SEMESTER IV

GRAPHIC ERA (DEEMED TO BE UNIVERSITY), DEHRADUN

SEMESTER IV

Name of Department: - Computer Science and Engineering

1. Subject Code: **TCS408** Course Title: **Programming in JAVA**
2. Contact Hours: **L:** **3** **T**: **0** **P**: **0**
3. Examination Duration (Hrs): **Theory** **3** **Practical** **0**
4. Relative Weight: **CIE** **25** **MSE** **25** **ESE** **50**
5. Credits: **3**
6. Semester: **IV**
7. Category of Course: **DSC**

8. Pre-requisite: Fundamental of Computer & Introduction to Programming (TCS 101), Programming for problem solving(TCS 201), Data Structures with C(TCS 302), Object Oriented Programming with C++ (TCS 307)

9. Course Outcome:

	After completion of the course the students will be able to**:** CO1 Explain the Java programming features and develop programs to demonstrate the same. CO2 Make use of object oriented concepts to develop applications CO3 Classify exceptions and demonstrate applications for file handling and multithreading. CO4 Analyze collection framework and develop applications using GUI CO5 Compare and utilize collection framework for programming application CO6 Design applications for event handling and accessing databases using Java features.

10. **Details of the Course:**

UNIT	CONTENTS	Contact Hrs
Unit - I	**Introduction to Java :**Importance and features of Java, Concepts of Java Virtual machine (JVM) Keywords, Constants, Variables and data types, operators and expressions, Control statements, Conditional statements, loops and iterations, Wrapper classes, Scanner Class: Scanner class methods (next(), nextLine() etc. **Concept of class:** Class definition, adding variables and methods, creating objects, constructors, defining methods, calling methods, Arrays, String Handling in java(String, StringBuffer classes)	10
Unit - II	**Object Oriented Programming concepts:** Inheritance, super classes, multilevel hierarchy, abstract and final classes, overloading and overriding **Packages and interfaces**: Packages, Defining Packages, Using Packages, import and static import, Access protection. **Interface:** Defining Interfaces, abstract methods declarations, implementing interfaces, extended interfaces, interface references.	9
Unit – III	**Exception handling:** Exception Types, Exception class, Runtime Exception Class, Error Class, Checked and unchecked Exceptions, Defining new exceptions; Handling: try, catch and finally; throw statement, throws clause. **Input/Output:** Basics, Byte and Character Streams, reading and writing from console and file. **Multithreaded programming:** Java thread model, synchronization, messaging, thread class, Runnable interface, inter thread communication, Producer/ consumer problems, Wait () and notify ().	9
Unit – IV	**Collection and Generic Framework:** Introduction to Collection and Generic Framework: Interfaces Iterator, List, Set, Array List, Linked List Hash Set and Array Deque classes **AWT &Swing:** Introduction to AWT and Swings, Swings advantages over AWT, Swing applications, Swing Controls : JButton ,JLabel , JCheckBox , JRadioButton , JList , JComboBox, JTextFiled, JTextArea , JScrollBar, JTable, Graphics in swing	9

Unit – V	**Event Handling:** Event delegation model, classes, Event Listener Interfaces,Adapter classes. **Java Database Connectivity (JDBC):** The Concept of JDBC, JBDC drivers (Type1 Driver,Type4 Driver), Connection interface, Statement interface, ResultSet interface, Creating and executing SQL statements.	9
	Total	**46**

Text Books:

Authors Name	Title	Edition	Publisher, Country	Year
Herbert Schildt	Java 2 The Complete Reference	9th Edition	McGraw Hill Education	2017
E. Balaguruswamy	Programming with Java- a Primer	6th Edition	McGraw Hill Education	2019

Reference Books:

Authors Name	Title	Edition	Publisher, Country	Year
Kathy Sierra, Bert Bates, Trisha Gee	Head First Java: A Brain-Friendly Guide	3rd Edition	O'Reilly Media, Inc.	2022
Cay S. Horstmann	Core Java, Volume I (Fundamentals) and Volume II	12th Edition	Addison-Wesley Professional	2021
Cay S. Horstmann	Core Java Volume II (Advanced Features)	12th Edition	Oracle Press	2021

Course Articulation Matrix

CO	Statement	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
TCS408.1	Explain the Java programming features and develop programs to demonstrate the same.	1	1	-	-	-	-	-	-	1	2	-	1	2	1	3
TCS408.2	Make use of object oriented concepts to develop applications	1	2	-	-	-	-	-	-	1	2	1	1	2	1	2
TCS408.3	Classify exceptions and demonstrate applications for file handling and multithreading.	-	2	-	-	2	1	-	-	1	2	-	-	3	2	2
TCS408.4	Analyse collection framework and develop applications using GUI.	-	3	1	3	2	-	-	-	1	1	1	1	3	2	2
TCS408.5	Compare and utilize collection framework for programming applications	2	2	-	1	1	1	-	-	1	1	1	1	3	1	2
TCS408.6	Design applications for event handling and accessing databases using Java features.	-	1	3	1	2	-	-	-	1	-	1	2	3	2	2
TCS 408		1.33	1.83	2	1.66	1.75	1	-	-	1	1.6	1	1.2	2.66	1.5	2.16

High correlation (3); Medium correlation (2); Low correlation (1), No correlation (-)

GRAPHIC ERA (DEEMED TO BE UNIVERSITY), DEHRADUN

SEMESTER IV

Name of Department: - Computer Science and Engineering

1. Subject Code: **TCS 402** Course Title: **Finite Automata and Formal Languages**
2. Contact Hours: **L:** **3** **T**: **0** **P**: **0**
3. Examination Duration (Hrs): **Theory** **3** **Practical** **0**
4. Relative Weight: **CIE** **25** **MSE** **25** **ESE** **50**
5. Credits: **3**
6. Semester: **IV**
7. Category of Course: **DSC**
8. Pre-requisite: Engineering Mathematics-I (TMA 101), Engineering Mathematics-II (TMA 201)

9. Course Outcome:	After completion of the course the students will be able to**:** CO1: Demonstrate the conversion of NFA into DFA, ϵ-NFA into DFA and Minimization of Finite Automata by using Myhill-Nerode Theorem CO2: Formulate DFA, RE and FA with output. CO3: Design CFG and check the language is not CFL. CO4: Design PDA and convert n-PDA into d-PDA. CO5: Design Turing machines for addition, subtraction, multiplication etc. CO6:Formulate finite machines; push down automata and Turing machines for automated functioning of devices.

10.**Details of the Course:**

UNIT	**CONTENTS**	**Contact Hrs**
Unit – I	Introduction; Alphabets, Strings and Languages; Automata and Grammars, Deterministic finite Automata (DFA)-Formal Definition, Simplified notation: State transition graph, Transition table, Language of DFA, Nondeterministic finite Automata (NFA), NFA with epsilon transition, Language of NFA, Equivalence of NFA and DFA, Minimization of Finite Automata, Distinguishing one string from other, Myhill-	10

	Nerode Theorem	
Unit - II	Regular expression (RE), Definition, Operators of regular expression and their precedence, Algebraic laws for Regular expressions, Kleen's Theorem, Regular expression to FA, DFA to Regular expression, Arden Theorem, Non Regular Languages, Pumping Lemma for regular Languages. Application of Pumping Lemma, Closure properties of Regular Languages, Decision properties of Regular Languages, FA with output: Moore and Mealy machine, Equivalence of Moore and Mealy Machine, Applications and Limitation of FA.	10
Unit – III	Context free grammar (CFG) and Context Free Languages (CFL): Definition, Examples, Derivation, Derivation trees, Ambiguity in Grammar, Inherent ambiguity, Ambiguous to Unambiguous CFG, Useless symbols, Simplification of CFGs, Normal forms for CFGs: CNF and GNF, Closure proper ties of CFLs, Decision Properties of CFLs: Emptiness, Finiteness and Membership, Pumping lemma for CFLs.	9
Unit – IV	Push Down Automata (PDA): Description and definition, Instantaneous Description, Language of PDA, Acceptance by Final state, Acceptance by empty stack, Deterministic PDA, Equivalence of PDA and CFG, CFG to PDA and PDA to CFG, Two stack PDA.	10
Unit – V	Turing machines (TM): Basic model, definition and representation, Instantaneous Description, Language acceptance by TM, Variants of Turing Machine, TM as Computer of Integer functions, Universal TM, Church's Thesis, Recursive and recursively enumerable languages, Halting problem, Introduction to Undecidability, Undecidable problems about TMs. Post correspondence problem (PCP), Modified PCP, Introduction to recursive function theory.	8
	Total	**47**

Text Books:

Authors Name	Title	Edition	Publisher, Country	Year
J. Hopcroft, R. Motwani, and J. Ullman.	Introduction to Automata Theory, Languages, and Computation,	3rd Edition	Pearson Education India	2008
KLP Mishra and N. Chandrasekaran,	Theory of Computer Science: Automata, Languages and Computation	3rd Edition	Prentice Hall Of India	2007

Reference Books:

Authors Name	Title	Edition	Publisher, Country	Year
Michael Sipser	Introduction to the Theory of Computation	3rd Edition	PWS Publishing Company	2012
Peter Linz	Introduction to Formal Languages and Automata	6th Edition	Jones and Bartlett Publishers, Inc.	2016

Course Articulation Matrix

CO	Statement	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
TCS402.1	Demonstrate the conversion of NFA into DFA, ϵ-NFA into DFA and Minimization of Finite Automata by using Myhill-Nerode Theorem	3	1	-	-	-	-	-	-	-	-	-	1	-	1	2
TCS402.2	Formulate DFA, RE and FA with output.	2	2	-	1	-	-	-	-	-	-	-	-	-	2	1
TCS402.3	Design CFG and check the language is not CFL.	1	1	-	2	1	-	-	1	-	-	-	2	2	1	-
TCS402.4	Design PDA and convert n-PDA into d-PDA.	3	1	-	1	-	-	-	1	-	-	-	1	1	2	1
TCS402.5	Design Turing machines for addition, subtraction, multiplication etc.	3	2	-	-	1	-	-	1	-	-	1	1	2	1	1
TCS402.6	Formulate finite machines; push down automata and Turing machines for automated functioning of devices.	3	2	-	2	1	-	-	1	-	-	1	1	1	1	1
TCS 402		2.5	1.5	-	1.5	1	-	-	1	-	-	1	1.2	1.5	1.33	1.2

High correlation (3); Medium correlation (2); Low correlation (1), No correlation (-)

GRAPHIC ERA (DEEMED TO BE UNIVERSITY), DEHRADUN

SEMESTER IV

Name of Department: - Computer Science and Engineering

1. Subject Code: **TCS403** Course Title: **Microprocessors**
2. Contact Hours: **L:** **3** **T**: **0** **P**: **0**
3. Examination Duration (Hrs): **Theory** **3** **Practical** **0**
4. Relative Weight: **CIE** **25** **MSE** **25** **ESE** **50**
5. Credits: **3**
6. Semester: **IV**
7. Category of Course: **DSC**
8. Pre-requisite: Basic Electronics Engineering(TEC 101 / TEC201), Fundamental of Computer & Introduction to Programming (TCS 101), TCS 301

9. Course Outcome:	After completion of the course the students will be able to**:** CO1 Identify of 8085 and 8086 microprocessors and memory segmentation CO2 Analysis of Instruction set of 8085and 8086. CO3 Implementation of different programs on 8085 and 8086 based microcomputer kit. CO4 Design the Interfacing of 8255 and 8085/8086. CO5 Design & develop Interfacing of microprocessor with Timing Devices1 CO6 Evaluate & Develop projects on embedded system using the foundation of microprocessor

10. **Details of the Course:**

UNIT	**CONTENTS**	**Contact Hrs**
Unit – I	Introduction to Microprocessors: Evolution of Microprocessors, Microcomputer , different type of buses, Example of an 8085 based System, Microprocessor Internal Architecture, Pin diagram and function of each pin, memory interfacing.	9
Unit - II	Programming with 8085: Instruction set, programming model of 8085, addressing modes, assembly language	10

	programming, Timing and control, peripheral I/O, memory mapped I/O, 8085 Interrupts, Stack and subroutines, Machine & Instruction cycle of 8085.	
Unit – III	16 Bit Processor: 16-bit Microprocessors (8086): Architecture, pin diagram, Physical address, segmentation, memory organization, Bus cycle, Addressing modes, Instruction set ,Assembly Language Programming of 8086, comparison of 8086 & 8088	8
Unit – IV	Interfacing (Data Transfer) with Microprocessor: Data Transfer Schemes: Introduction, handshaking signals, Types of transmission, 8255 (PPI), Serial Data transfer (USART 8251), memory interfacing, 8257 (DMA), programmable interrupt Controller (8259).	9
Unit – V	Interfacing of Microprocessor with Timing Devices: Programmable Interval Timer/ Counter (8253/8254): Introduction, modes, Interfacing of 8253, applications, Need of ADC & DAC, resolution, Introduction to DAC & ADC, ADC & DAC Interfacing (0808, 0809).	9
	Total	**45**

Text Books:

Authors Name	Title	Edition	Publisher, Country	Year
Ramesh Gaonkar	Microprocessor Architecture, Programming, and Applications with the 8085	6th	Penram International Publication (India) Pvt. Ltd	2013
A. K. Ray & K. M. Bhurchandi	Advanced Microprocessors and peripherals	3rd	Tata McGraw Hill	2012
Muhammad Ali Mazidi, Janice Gillispie Mazidi,	8051 Microcontroller & Embedded System,	2nd	Pearson / PHI publication	2007

Reference Books:

Authors Name	Title	Edition	Publisher, Country	Year
Douglas V. Hall,	Microprocessors and Interfacing,	3rd	Tata McGraw Hill	2012
Barry B. Brey,	The Intel Microprocessors Architecture Programming and interfacing,	8th	Pearson	2012

Course Articulation Matrix

CO	Statement	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
TCS403.1	Identify of 8085 and 8086 microprocessors and memory segmentation	1	1	-	1	2	-	-	-	1	2	-	1	2	1	2
TCS403.2	Analysis of Instruction set of 8085and 8086.	2	3	-	-	1	-	-	-	1	-	-	-	3	2	2
TCS403.3	Implementation of different programs on 8085 and 8086 based microcomputer kit.	1	1	-	3	-	-	-	-	2	-	1	2	3	2	2
TCS403.4	Design the Interfacing of 8255 and 8085/8086.	1	1	3	2	3	-	-	-	2	1	-	-	3	1	2
TCS403.5	Design & develop Interfacing of microprocessor with Timing Devices	3	2	3	-	1	-	-	-	1	-	-	-	3	1	1
TCS403.6	Evaluate & Develop projects on embedded system using the foundation of microprocessor	1	1	2	2	-	2	-	2	3	3	2	s	2	1	2
TCS403		1.5	1.5	2	1.66	1.75	2	-	2	1.8	2	1.5	1	2.66	1.66	1.83

High correlation (3); Medium correlation (2); Low correlation (1), No correlation (-)

GRAPHIC ERA (DEEMED TO BE UNIVERSITY), DEHRADUN

SEMESTER IV

Name of Department: - Computer Science and Engineering

1. Subject Code: **TCS409** Course Title: **Design and Analysis of Algorithms**
2. Contact Hours: **L:** **3** **T**: **0** **P**: **0**
3. Examination Duration (Hrs): **Theory** **3** **Practical** **0**
4. Relative Weight: **CIE** **25** **MSE** **25** **ESE** **50**
5. Credits: **3**
6. Semester: **IV**
7. Category of Course: **DSC**
8. Pre-requisite: Fundamental of Computer & Introduction to Programming(TCS101), Programming for problem solving(TCS201), Data Structures with C(TCS302)

9. Course Outcome:	After completion of the course the students will be able to**:** CO1 Discuss various asymptotic notations to analyse time and space complexity of algorithms CO2 Analyse the various paradigms for designing efficient algorithms using concepts of design and conquer, greedy and dynamic programming techniques CO3 Provide solutions to complex problems using the concept of back tracking and branch and bound techniques. CO4 Apply algorithm design techniques to predict the complexity of certain NP complete problems. CO5 Implement Dijkstra's, Bellman-ford, Prims, Kruskal's algorithms to solve the real world problems like traveling salesman problem, job sequencing, packet routing etc. CO6 Apply pattern matching algorithms like Rabin Karp Algorithm, Brute-force techniques etc, to find a particular pattern.

10. **Details of the Course:**

UNIT	CONTENTS	Contact Hrs
Unit – I	**Asymptotic Notations and Searching Algorithms** **Introduction to Algorithms -** What is an Algorithm, Rate of growth, Commonly used rate of growths, Types	8

	of analysis, Asymptotic Notations, Master theorem **Searching -** Linear search (sorted and unsorted), Iterative and recursive binary search, Exponential search, Tower of Hanoi and solving its recursion, Fibonacci and solving its recursion	
Unit - II	**Sorting Algorithms Sorting -** Bubble sort, Insertion sort, selection sort, quick sort, randomized quick sort, merge sort, Heap & Heap sort , counting sort, External sorting, Radix sort , bucket sort. **Divide sorting algorithms into following types -** online sort, stable sort, in place sort, Comparison of sorting algorithms on the basis of number of swaps, by number of comparisons, recursive or iterative nature, time and space complexity	10
Unit – III	**Graph Algorithms** Representation of Graphs, Breadth-first search (BFS), depth-first search (DFS), topological sort, Difference between BFS and DFS **Data structures for disjoint sets -** Finding cycle in a graph, Finding strongly connected components **Minimum spanning trees -** Kruskal and Prim algorithms (Greedy Algorithms) **Single source shortest paths -** Dijkstra (Greedy Approach) and Bellman ford (Dynamic Programming) algorithms, Working on -ve edge & cycle, difference & similarity. **All pair shortest paths -** The Floyd Warshall algorithm	12
Unit – IV	**Algorithm Design Techniques - Greedy and Dynamic Programming** **Greedy algorithms –O**ptimal substructure property, Activity selection problem, Job sequencing problem, Huffman codes, fractional knapsack problem **Dynamic Programming -** Overlapping substructure property, Optimal substructure property, Tabulation vs Memorization, Fibonacci numbers, 0/1 Knapsack problem, Longest common subsequence, Matrix chain multiplication, Longest increasing subsequence.	10

Unit – V	**Hashing, String Matching and NP-Completeness** **Hashing -** Introduction to Hashing, Hash function, Collision and collision handling, - Chaining, Open addressing (longest probing, quadratic probing, double hashing) **String Matching -** Naive string-matching algorithm, The Rabin-Karp algorithm, The Knuth-Morris-Pratt algorithm,Trie. **NP-Completeness** - Importance of NP-completeness, P, NP, NP Complete and NP hard problems, Polynomial time and polynomial time verification, The subset-sum problem, The traveling salesman problem	10
	Total	**50**

Text Books:

Authors Name	Title	Edition	Publisher, Country	Year
Thomas H. Cormen, Charles E. Leiserson, Ronal L. Rivest, and Clifford Stein	Introduction to Algorithms	4th Edition	MIT Press	2022

Reference Books:

Authors Name	Title	Edition	Publisher, Country	Year
Donald Knuth	Art of Computer Programming, The: Volume 1: Fundamental Algorithms (ART OF COMPUTER PROGRAMMING)	3rd Edition	Addison-Wesley	1998

Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran:"	Fundamentals of Computer Algorithms	2nd Edition	Universities press	2007
Anany Levitin	Introduction to the Design & Analysis of Algorithms	2nd Edition	Pearson Education	2008

Course Articulation Matrix

CO	Statement	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
TCS409.1	Discuss various asymptotic notations to analyse time and space complexity of algorithms	-	1	-	1	-	-	-	-	1	1	-	1	2	2	2
TCS409.2	Analyse the various paradigms for designing efficient algorithms using concepts of design and conquer, greedy and dynamic programming techniques	1	3	-	1	1	-	1	-	1	1	-	2	3	2	2
TCS409.3	Provide solutions to complex problems using the concept of back tracking and branch and bound techniques.	-	-	3	3	2	-	-	1	-	1	1	2	2	2	2
TCS409.4	Apply algorithm design techniques to predict the complexity of certain NP complete problems.	3	-	1	1	1	-	-	1	-	1	-	1	2	3	2
TCS409.5	Implement Dijkstra's, Bellman-ford, Prims, Kruskal's algorithms to solve the real world problems like traveling salesman problem, job sequencing, packet routing etc.	1	1	2	3	-	-	-	-	1	2	-	1	3	2	1
TCS409.6	Apply pattern matching algorithms like Rabin Karp Algorithm, Brute-force techniques etc., to find a particular pattern.	1	3	1	-	-	-	-	-	1	1	-	1	2	3	2
TCS 409		1.5	2	1.75	1.8	1.33	-	1	1	1	1.16	1	1.33	2.33	2.33	1.83

High correlation (3); Medium correlation (2); Low correlation (1), No correlation (-)

GRAPHIC ERA (DEEMED TO BE UNIVERSITY), DEHRADUN

SEMESTER IV

Name of Department: - Computer Science and Engineering

1. Subject Code: **PCS408** Course Title: **Java Programming Lab**
2. Contact Hours: **L:** **0** **T**: **0** **P**: **2**
3. Examination Duration (Hrs): **Theory** **0** **Practical** **3**
4. Relative Weight: **CIE** **25** **MSE** **25** **ESE** **50**
5. Credits: **1**
6. Semester: **4**
7. Category of Course: **DSC**
8. Pre-requisite: OOPS with C++ Lab (PCS307), Object oriented Programming with C++ (TCS307)

9. Course Outcome:	After completion of the course the students will be able to**:** CO1 Understand the object-oriented approach in programming along with the purpose and usage principles of inheritance, polymorphism, encapsulation, and method overloading etc. CO2 Demonstrate ability to test and debug Java programs using IDE. CO3 Analyse, design, and develop small to medium sized application programs that demonstrate professionally acceptable programming standards. CO4 Demonstrate skills of developing event-driven programs using graphical user interfaces. CO5 Demonstrate skills of developing event-driven programs using graphical user interfaces.

10. **Details of the Course:**

Sl. No.	List of problems for which student should develop program and execute in the Laboratory	Contact Hours
1.	**Taking input from Command line and convert objects into primitivedata type:** Write a java program to take input as a command line argument. Your name, course, university rollno and semester. Display the information. Name: University RollNo:Course:	2

	Semester:	
2.	**Concepts of Java Control statements, Conditional statements, loops and iterations, Wrapper classes, Scanner Class:** Using the switch statement, write a menu-driven program to calculate the maturity amount of a bank deposit. The user is given the following options: (i) Term Deposit (ii) Recurring Deposit For option (i) accept Principal (p), rate of interest (r) and time period in years (n). Calculate and output the maturity amount (a) receivable using the formula a = p[1 + r / 100]n. For option (ii) accept monthly instalment (p), rate of interest (r) and time period in months (n). Calculate and output the maturity amount (a) receivable using the formula a = p * n + p * n(n + 1) / 2 * r / 100 * 1 / 12. For an incorrect option, an appropriate error message should be displayed. [Use Scanner Class to take input]	2
3.	Program to find if the given numbers are Friendly pair or not (Amicable or not). **Friendly Pair** are two or more numbers with a common abundance. **Input & Output format:** • Input consists of **2 integers**. • The first integer corresponds to number 1 andthe second integer corresponds to number 2. If it is a Friendly Pair display **Friendly Pair** or displays Not Friendly Pair. For example,6 and 28 are Friendly Pair. (Sum of divisors of 6)/6 = (Sum of divisors of 28)/28. Steps to check whether the given numbers are friendly pair or not ▪ Input the numbers num1 and num2. ▪ Initialize sum1 = sum2 = 0. ▪ sum1 = sum of all divisors of num1. ▪ sum2 = sum of all divisors of num2. ▪ If (sum1 == num1) and (sum2 == num2), then print "Abundant Numbers".	2

	▪ Else, print "Not Abundant Numbers". Program to check whether the given numbers are friendly pair or not	
4.	Program to replace all 0's with 1 in a given integer. Given an integer as an input, all the 0's in the number has to be replaced with 1. For example, consider the following number Input: 102405 **Output: 112415** **Input: 56004** **Output: 56114** Steps to replace all 0's with 1 in a given integer • Input the integer from the user. • Traverse the integer digit by digit. • If a '0' is encountered, replace it by '1'. • Print the integer.	2
5.	**Array in Java:** Printing an array into Zigzag fashion. Suppose youwere given an array of integers, and you are told to sort the integers in a zigzag pattern. In general, in a zigzag pattern, the first integer is less than the second integer, which is greater than the third integer, which is less than the fourth integer, and so on. Hence, the converted array should be in the form of e1 < e2 > e3 < e4 > e5 < e6. **Test cases: Input 1:** 7 4 3 7 8 6 2 1 **Output 1:** 3 7 4 8 2 6 1 **Input 2:** 4 1 4 3 2 **Output 2:** 1 4 2 3	2
6.	The problem to rearrange positive and negative numbers in an array .	2

	Method: This approach moves all negative numbers to the beginning and positive numbers to the end butchanges the order of appearance of the elements of the array. Steps: 1. Declare an array and input the array elements. 2. Start traversing the array and if the current element is negative, swap the current elementwith the first positive element and continue traversing until all the elements have been encountered. 3. Print the rearranged array. Test case: • Input: 1 -1 2 -2 3 -3 Output: -1 -2 -3 1 3 2	
7.	Program to find the saddle point coordinates in a given matrix. A saddle point is an element of the matrix, which is the minimum element in its row and the maximum in its column. For example, consider the matrix given belowMat [3][3] 1 2 3 4 5 6 7 8 9 Here, 7 is the saddle point because it is the minimum element in its row and maximum element in its column. Steps to find the saddle point coordinates in a givenmatrix. 1. Input the matrix from the user. 2. Use two loops, one for traversing the row andthe other for traversing the column. 3. If the current element is the minimum element inits row and maximum element in its column, then return its coordinates. Else, continue traversing.	2
8.	**String Handling in Java (using String and StringBuffer class):** Program to find all the patterns of 0(1+)0 in the given string. Given a string containing 0's and 1's, find the total number of 0(1+)0 patterns in the string and output it. 0(1+)0 - There should be at least one '1' between the two 0's. For example, consider the following string.	2

	Input: 01101111010 **Output:** 3 **Explanation: 0110**1111010 - count = 1	
9.	Write a java program to delete vowels from given string using StringBuffer class	2
10	**Class definition, creating objects and constructors:** Write a java program to create a class named 'Bank ' with the following data members: • Name of depositor • Address of depositor • Account Number • Balance in account **Class 'Bank' has a method for each of the following:** 1. Generate a unique account number for each depositor. 2. For first depositor, account number will be 1001, for second depositor it will be 1002 and so on 3. Display information and balance of depositor 4. Deposit more amount in balance of any depositor 5. Withdraw some amount from balance deposited. 6. Change address of depositor **After creating the class, do the following operations.** 1. Enter the information (name, address, account number, balance) of the depositors. Number of depositors is to be entered by the user. 2. Print the information of any depositor. 3. Add some amount to the account of any depositor and then display final information of that depositor. 4. Remove some amount from the account of any depositor and then display final information of that depositor. 5. Change the address of any depositor and then display the final	2

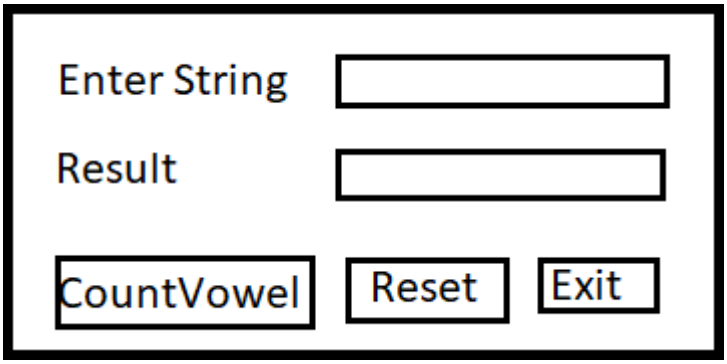
	information of that depositor. 6. Randomly repeat these processes for some other bank accounts.	
11	Define a class **Word Example** having the followingdescription: Data members/instance variables: Private String strdata: to store a sentence. **Parameterized Constructor WordExample(String)** : Accept a sentence which may be terminated by either'.', '? 'or'!' only. The wordsmay be separated by more than one blank space and are in UPPER CASE. **Member Methods:** **void countWord():** Find the number of wordsbeginning and ending with a vowel. **void placeWord():** Place the words which begin andend with a vowel at the beginning, followed by the remaining words as they occur in the sentence	2
12	**Method overloading (Compile time Polymorphism):** Write a Java program to create a class called **ArrayDemo** and overload arrayFunc() function. **void arrayFunc(int [], int)** □ To find all pairs of elements in an Array whose sum is equal to a givennumber : Array numbers= [4, 6, 5, -10, 8, 5, 20], target=10 **Output :** Pairs of elements whose sum is 10 are :4 + 6 = 10 5 + 5 = 10 -10 + 20 = 10 **void arrayFunc(int A[], int p, int B[], int q)** □ Giventwo sorted arrays A and B of size p and q, Overload method arrayFunc() to merge elements of A with B bymaintaining the sorted order i.e. fill A with first p smallest elements and fill B with remaining elements. Example: Input :	2

	int[] A = { 1, 5, 6, 7, 8, 10 } int[] B = { 2, 4, 9 } Output: Sorted Arrays: A: [1, 2, 4, 5, 6, 7] B: [8, 9, 10] **(Use Compile time Polymorphism MethodOverloading)**	
13	**Method overriding (Runtime Polymorphism), Abstract class and Abstract method, Inheritance, interfaces:** Write a java program to calculate the area of a rectangle, a square and a circle. Create an abstract class 'Shape' with three abstract methods namely rectangleArea() taking two parameters, squareArea() and circleArea() taking one parameter each. Now create another class 'Area' containing all the three methods rectangleArea(), squareArea() and circleArea() for printing the area of rectangle, square and circle respectively. Create an object of class Area and call all the three methods. **(Use Runtime Polymorphism)**	2
14	Write a java program to implement abstract class andabstract method with following details: **Create a abstract Base Class TemperatureData members:** double temp; **Method members:** void setTempData(double) abstact void changeTemp() **Sub Class Fahrenheit (subclass of Temperature) Data members**: double ctemp; **method member:** Override abstract method changeTemp() to convert Fahrenheit temperature into degree Celsius by using formula C=5/9*(F-32) and display converted temperature Sub Class Celsius (subclass of Temperature) **Data member:**	2

	double ftemp; **Method member:** Override abstract method changeTemp() to convert degree Celsius into Fahrenheit temperature by using formula F=9/5*c+32 and display converted temperature	
15	Write a java program to create an interface that consists of a method to display **volume ()** as an abstract method and redefine this method in the derived classes to suit their requirements. Create classes called **Cone**, **Hemisphere** and **Cylinder** that implements the interface. Using these three classes, design a program that will accept dimensions of a cone, cylinder and hemisphere interactively and display the volumes. Volume of cone = (1/3)πr2h Volume of hemisphere = (2/3)πr3Volume of cylinder = πr2h	2
16	Write a java program to accept and print the employee details during runtime. The details will include employee id, name, dept_ Id. The program should raise an exception if user inputs incomplete or incorrect data. The entered value should meet the following conditions: a. First Letter of employee name should be in capital letter. b. Employee id should be between 2001 and 5001 c. Department id should be an integer between 1 and 5. If the above conditions are not met, then the application should raise specific exception else should complete normal execution.	2
17	Create a class MyCalculator which consists of a single method power (int, int). This method takes two integers, n and p, as parameters and finds np. If either n or p is negative, then the method must throw an exception which says, "n and p should be non- negative". **Input Format** Each line of the input contains two integers, n and p. Output Format Each line of the output contains the result, if neither of n and p is negative. Otherwise, the output contains "n and p should be non- negative".	2

	Sample Input 3 5 2 4 0 0 -1 -2 -1 3 **Sample Output** 243 16 java.lang.Exception: n and p should not be zero. java.lang.Exception: n or p should not be negative. java. lang. Exception: n or p should not be negative. **Explanation** In the first two cases, both *n* and *p* are positive. So, the power function returns the answer correctly. In the third case, both *n* and *p* are zero. So, the exception, "n and p should not be zero." is printed. In the last two cases, at least one out of n and p is negative. So, the exception, "n or p should not be negative." is printed for these two cases.	
18	**File Handling in Java:** Write a java file handling program to count and displaythe number of palindromes present in a text file "myfile.txt". Example: If the file "myfile.txt" contains the following lines, My name is NITIN Hello aaa and bbb wordHow are You ARORA is my friendOutput will be => 4	2
19	**Multithreaded programming**: Write a program MultiThreads that creates two threads-one thread with the name CSthread and the other thread named ITthread. Each thread should display its respective name and execute after a gap of 500 milliseconds. Each thread should also display a number indicating the number of times it got a chance to execute.	
20.	Write a java program for to solve producer consumer problem in which a producer produces a value and consumer consume the value	2

	before producergenerate the next value	
21	**Collection and Generic Framework:** Write a method removeEvenLength that takes an ArrayList of Strings as a parameter and that removes all the strings of even length from the list. **(Use ArrayList)**	
22	Write a method swapPairs that switches the order of values in an ArrayList of Strings in a pairwise fashion. Your method should switch the order of the first two values, then switch the order of the next two, switch the order of the next two, and so on. For example, if the list initially stores these values: {"four", "score", "and", "seven", "years", "ago"} your method should switch the first pair, "four", "score", the second pair, "and", "seven", and the third pair, "years", "ago", to yield this list: {"score", "four", "seven", "and", "ago", "years"} If there are an odd number of values in the list, the final element is not moved. For example, if the original list had been: {"to", "be", "or", "not", "to", "be", "hamlet"} It would again switch pairs of values, but the final value, "hamlet" would not be moved, yielding this list: {"be", "to", "not", "or", "be", "to", "hamlet"}	2
23	Write a method called alternate that accepts two Listsof integers as its parameters and returns a new List containing alternating elements from the twolists, in the following order: • First element from first list • First element from second list • Second element from first list • Second element from second list • Third element from first list • Third element from second list If the lists do not contain the same number of elements, the remaining elements from the longer list should be placed consecutively at the end. For example, for a first list of (1, 2, 3, 4, 5) and a second list of (6, 7, 8, 9, 10, 11, 12), a call of alternate (list1,	2

	list2) should return a list containing (1, 6, 2, 7, 3, 8, 4, 9, 5, 10, 11, 12). Do not modify the parameter lists passed in.	
24	**AWT & Swing, Event Handling:** Write a GUI program to develop an application that receives a string in one text field, and count number ofvowels in a string and returns it in another text field, when the button named “CountVowel” is clicked. When the button named “Reset” is clicked it will resetthe value of textfield one and Textfield two. When the button named “Exit” is clicked it will closed the application. 	2
25	**Java Database Connectivity (JDBC):** **Create a database of employee with the following fields.** • **Name** • **Code** • **Designation** • **Salary**	2

	 a) Write a java program to create GUI java application to take employee data from the TextFields and store it in database using JDBC connectivity. b) Write a JDBC Program to retrieve all the records from the employee database.	
	Total	**48**

Text Books:

Authors Name	Title	Edition	Publisher, Country	Year
Herbert Schildt	Java 2 The Complete Reference	9TH Edition	McGraw Hill Education	2017
E. Balaguruswamy	Programming with Java- a Primer	6TH Edition	McGraw Hill Education	2019

Reference Books:

Authors Name	Title	Edition	Publisher, Country	Year
Kathy Sierra, Bert Bates, Trisha Gee	Head First Java: A Brain-Friendly Guide	3rd Edition	O'Reilly Media, Inc.	2022
Cay S. Horstmann	Core Java, Volume I (Fundamentals) and Volume II	12th Edition	Addison-Wesley Professional	2021
Cay S. Horstmann	Core Java Volume II (Advanced Features)	12th Edition	Oracle Press	2021

Course Articulation Matrix

CO	Statement	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
PCS408.1	Understand the object-oriented approach in programming along with the purpose and usage principles of inheritance, polymorphism, encapsulation, and method overloading etc.	1	1	-	-	1	-	-	-	-	2	-	2	2	1	2
PCS408.2	Demonstrate ability to test and debug Java programs using IDE.	1	2	-	1	3	-	-	-	1	3	1	2	2	3	2
PCS408.3	Analyse, design, and develop small to medium sized application programs that demonstrate professionally acceptable programming standards	1	3	3	1	2	-	-	1	1	-	3	2	3	2	2
PCS408.4	Demonstrate skills of developing event-driven programs using graphical user interfaces.	-	1	1	1	1	-	-	1	-	2	-	1	2	1	3
PCS 408		1	1.75	2	1	1.75	-	-	1	1	2.33	2	1.75	2.25	1.75	2.25

High correlation (3); Medium correlation (2); Low correlation (1), No correlation (-)

GRAPHIC ERA (DEEMED TO BE UNIVERSITY), DEHRADUN

SEMESTER IV

Name of Department: - Computer Science and Engineering

1. Subject Code: **PCS403** Course Title: **Microprocessors lab**
2. Contact Hours: **L:** **0** **T**: **0** **P**: **2**
3. Examination Duration (Hrs): **Theory** **0** **Practical** **3**
4. Relative Weight: **CIE** **25** **MSE** **25** **ESE** **50**
5. Credits: **1**
6. Semester: **IV**
7. Category of Course: **DSC**
8. Pre-requisite: Electronics Engineering Lab (PEC151 / PEC251), Logic design lab (PCS308)

9. Course Outcome:	After completion of the course the students will be able to**:** CO1 Remember 8085 and 8086 instruction set. CO2 Understand different assembly language programs on microprocessor-based microcomputer kit. CO3 Apply the programming concepts to test and debug assembly language programs in the laboratory. CO4 Assemble various devices and memories with microprocessor for any defined task.

10. **Details of the Course:**

Sl. No.	List of problems for which student should develop program and execute in the Laboratory	Contact Hours
1.	Write program in 8085 to swap two 8-bit numbers.	2
2.	Write a program in 8085 to move a block of data bytes from one location to another location.	1
3.	Write programs in 8085 to perform addition & subtraction of 8-bit number with carry / borrow.	1
4.	Write a program in 8085 for addition of 16 bits numbers with carry.	1
5.	Write a program for multiplication of two 8-bit numbers in 8085.	1
6.	Write an ALP in 8085 to add two 8-bit BCD data.	1
7.	(a) Write an ALP in 8085 to find larger number between two numbers. (b) Write an ALP in 8085 to find smaller number between two numbers.	2

8.	Write an ALP in 8085 to find largest /smallest in a series of n number.	1
9.	Write a program to find square root of a number in 8085.	1
10.	Write a program for division of two 8 bit numbers in 8085.	1
11.	Write a program in 8085 to count number of ones in an 8 bit number.	1
12.	Write a program in 8085 to find sum of digits of an 8 bit number.	1
13.	(a) Write a program in 8086 to add two 16-bit numbers given by the user. (b) Write a program in 8086 to subtract two 16-bit numbers given by the user.	2
14.	(a) Write a program in 8086 to multiply two 8-bit data. (b) Write a program in 8086 to divide: 8-bit data by 8-bit data.	2
15.	Write a program in 8086 to find the largest no. from an array of n numbers stored in an array.	1
16.	Write a program in 8086 to add and subtract two 8-bit BCD numbers.	1
17.	Write a program in 8086 to convert a BCD number to its Binary code equivalent.	1
18.	Write a program in 8086 to perform sorting of given set of numbers.	1
19.	Write a program in 8086 to convert a Binary number to its gray code equivalent.	1
20.	Write a program in 8086 to convert a BCD number to its ASCII code equivalent.	1
	Total	**24**

Text Books:

Authors Name	Title	Edition	Publisher, Country	Year
Ramesh Gaonkar	Microprocessor Architecture, Programming, and Applications with the 8085	6th Edition	Penram International Publication (India) Pvt. Ltd	2013
A. K. Ray & K. M.	Advanced Microprocessors and	3rd Edition	Tata McGraw Hill	2012

Bhurchandi	peripherals			
Muhammad Ali Mazidi, Janice Gillispie Mazidi,	8051 Microcontroller & Embedded System,	2nd Edition	Pearson / PHI publication	2007

Reference Books:

Authors Name	**Title**	**Edition**	**Publisher, Country**	**Year**
Douglas V. Hall,	Microprocessors and Interfacing,	3rd Edition	Tata McGraw Hill	2012
Barry B. Brey,	The Intel Microprocessors Architecture Programming and interfacing,	8th Edition	Pearson	2012

Course Articulation Matrix

CO	Statement	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
PCS403.1	Remember 8085 and 8086 instruction set.	1	1		1	2				2			1	1		1
PCS403.2	Understand different assembly language programs on microprocessor-based microcomputer kit.	1	2	3		2		1	2	2	2		2	2	2	
PCS403.3	Apply the programming concepts to test and debug assembly language programs in the laboratory.	2	2	2	2	2	1		2	2	2	1	2	2	2	1
PCS403.4	Assemble various devices and memories with microprocessor for any defined task.	2	2	2	2	2		1	2	2	2		2	2	3	2
PCS 403		1.5	1.75	2.33	1.66	2	1	1	2	2	2	1	1.75	1.75	2.33	1.33

High correlation (3); Medium correlation (2); Low correlation (1), No correlation (-)

GRAPHIC ERA (DEEMED TO BE UNIVERSITY), DEHRADUN

SEMESTER IV

Name of Department: - Computer Science and Engineering

1. Subject Code: **PCS409** Course Title: **Design and Analysis of Algorithms**
2. Contact Hours: **L:** **0** **T**: **0** **P**: **2**
3. Examination Duration (Hrs): **Theory** **0** **Practical** **3**
4. Relative Weight: **CIE** **25** **MSE** **25** **ESE** **50**
5. Credits: **1**
6. Semester: **IV**
7. Category of Course: **DSC**
8. Pre-requisite: Data Structures Lab (PCS302), Any programming language

9. Course Outcome:	After completion of the course the students will be able to**:** CO1 Analyse algorithmic time and space complexity using asymptotic notations. CO2 Design efficient algorithms utilizing techniques such as divide and conquer, greedy, and dynamic programming. CO3 Solve complex problems using backtracking and branch-and-bound techniques. CO4 Predict the complexity of NP-complete problems and propose algorithmic approaches. CO5 Apply Dijkstra's, Bellman-Ford, Prim's, and Kruskal's algorithms to real-world problems. CO6 Implement pattern matching algorithms like Rabin-Karp and brute-force techniques for pattern identification.

10. **Details of the Course:**

S. No.	List of problems for which student should develop program and execute in the Laboratory	Contact Hours
1.	Week 1: Note: Input, output format for problem I, II and III is same and is given at the end of this exercise. I. Given an array of nonnegative integers, design a linear algorithm and implement it using a program to find whether given key element is present in the array or not. Also, find total number of comparisons for	2

	each input case. (Time Complexity = O(n), where n is the size of input) Sample I/O Problem - 1: Input: Output: 3 Present 6 8 Present 3 34 35 65 31 25 89 64 30 Not Present 6 89 5 977 354 244 546 355 244 6 23 64 13 67 43 56 63 II. Given an already sorted array of positive integers, design an algorithm and implement it using a program to find whether given key element is present in the array or not. Also, find total number of comparisons for each input case. (Time Complexity = O(nlogn), where n is the size of input). III. Given an already sorted array of positive integers, design an algorithm and implement it using a program to find whether a given key element is present in the sorted array or not. For an array arr[n], search at the indexes arr[0], arr[2], arr[4],. ,arr[2k] and so on. Once the interval (arr[2k] < key < arr[2k+1]) is found, perform a linear search operation from the index 2k to find the element key. (Complexity < O(n), where n is the number of elements need to be scanned for searching): Jump Search Input format: The first line contains number of test cases, T. For each test case, there will be three input lines. First line contains n (the size of array). Second line contains n space-separated integers describing array. Third line contains the key element that need to be searched in the array. Output format: The output will have T number of lines.	

	For each test case, output will be “Present” if the key element is found in the array, otherwise “Not Present”. Also for each test case output the number of comparisons required to search the key. Sample I/O Problem - 2, 3: Input: Output: 3 Present 3 5 Not Present 4 12 23 36 39 41 Present 3 41 8 21 39 40 45 51 54 68 72 69 10 101 246 438 561 796 896 899 4644 7999 8545 7999	
2.	Week 2: I. Given a sorted array of positive integers containing few duplicate elements, design an algorithm and implement it using a program to find whether the given key element is present in the array or not. If present, then also find the number of copies of given key. (Time Complexity = O(log n)) Input format: The first line contains number of test cases, T. For each test case, there will be three input lines. First line contains n (the size of array). Second line contains space-separated integers describing array. Third line contains the key element that need to be searched in the array. Output format: The output will have T number of lines. For each test case T, output will be the key element and its number of copies in the array if the key element is present in the array otherwise print “ Key not present”. Sample I/O Problem I: Input: Output:	2

	2 981 - 2 10 75 - 3 235 235 278 278 763 764 790 853 981 981 981 15 1 2 2 3 3 5 5 5 25 75 75 75 97 97 97 75 II. Given a sorted array of positive integers, design an algorithm and implement it using a program to find three indices i, j, k such that arr[i] + arr[j] = arr[k]. Input format: The first line contains number of test cases, T. For each test case, there will be two input lines. First line contains n (the size of array). Second line contains space-separated integers describing array. Output: The output will have T number of lines. For each test case T, print the value of i, j and k, if found else print “No sequence found”. Sample I/O Problem II: Input: Output: 3 No sequence found. 5 2, 7, 8 1 5 84 209 341 1, 6, 9 10 24 28 48 71 86 89 92 120 194 201 15 64 69 82 95 99 107 113 141 171 350 369 400 511 590 666 III. Given an array of nonnegative integers, design an algorithm and a program to count the number of pairs of integers such that their difference is equal to a given key, K. Input format: The first line contains number of test cases, T. For each test case, there will be three input lines. First line contains n (the size of array).	

	Second line contains space-separated integers describing array. Third line contains the key element. Output format: The output will have T number of lines. For each test case T, output will be the total count i.e. number of times such pair exists. Sample I/O Problem III: Input: Output: 2 2 5 4 1 51 84 21 31 20 10 24 71 16 92 12 28 48 14 20 22 4	
3.	Week 3: I. Given an unsorted array of integers, design an algorithm and a program to sort the array using insertion sort. Your program should be able to find number of comparisons and shifts (shifts - total number of times the array elements are shifted from their place) required for sorting the array. Input Format: The first line contains number of test cases, T. For each test case, there will be two input lines. First line contains n (the size of array). Second line contains space-separated integers describing array. Output Format: The output will have T number of lines. For each test case T, there will be three output lines. First line will give the sorted array. Second line will give total number of comparisons. Third line will give total number of shift operations required. Sample I/O Problem I: Input: Output: 3 -31 -23 32 45 46 65 76 89 8 comparisons = 13	1

-23 65 -31 76 46 89 45 32 shifts = 20
10 21 32 34 46 51 54 65 76 78 97
54 65 34 76 78 97 46 32 51 21 comparisons = 28
15 shifts = 37
63 42 223 645 652 31 324 22 553 -12 54 65 86 46 325 -12 22 31 42 46 54 63 65 86 223 324 325 553 645 652
comparisons = 54
shifts = 68

II. Given an unsorted array of integers, design an algorithm and implement a program to sort this array using selection sort. Your program should also find number of comparisons and number of swaps required.

Input Format:
The first line contains number of test cases, T. For each test case, there will be two input lines. First line contains n (the size of array).
Second line contains space-separated integers describing array.

Output Format:
The output will have T number of lines.
For each test case T, there will be three output lines. First line will give the sorted array.
Second line will give total number of comparisons. Third line will give total number of swaps required.

Sample I/O Problem II:

Input:	Output:
3	-21 -13 12 45 46 65 76 89
8	comparisons = 28
-13 65 -21 76 46 89 45 12	swaps = 7
10	21 32 34 46 51 54 65 76 78 97
54 65 34 76 78 97 46 32 51 21	comparisons = 45
15	swaps = 9
63 42 223 645 652 31 324 22 553 12 54 65 86 46 325	12 22 31 42 46 54 63 65 86 223 324 325 553 645 652
	comparisons = 105
	swaps = 14

	III. Given an unsorted array of positive integers, design an algorithm and implement it using a program to find whether there are any duplicate elements in the array or not. (use sorting) (Time Complexity = O(n log n)) Input Format: The first line contains number of test cases, T. For each test case, there will be two input lines. First line contains n (the size of array). Second line contains space-separated integers describing array. Output Format: The output will have T number of lines. For each test case, output will be 'YES' if duplicates are present otherwise 'NO'. Sample I/O Problem III: Input: Output: 3 NO 5 YES 28 52 83 14 75 NO 10 75 65 1 65 2 6 86 2 75 8 15 75 35 86 57 98 23 73 1 64 8 11 90 61 19 20	
4.	Week 4: I. Given an unsorted array of integers, design an algorithm and implement it using a program to sort an array of elements by dividing the array into two subarrays and combining these subarrays after sorting each one of them. Your program should also find number of comparisons and inversions during sorting the array. Input Format: The first line contains number of test cases, T. For each test case, there will be two input lines. First line contains n (the size of array). Second line contains space-separated integers describing array. Output Format: The output will have T number of lines. For each test case T, there will be three output lines. First line will give the sorted array. Second line will give total number of comparisons. Third line will give	1

total number of inversions required.

Sample I/O Problem I:

Input: Output:

3 21 23 32 45 46 65 76 89

8 comparisons = 16

23 65 21 76 46 89 45 32 inversions =

10 21 32 34 46 51 54 65 76 78 97

54 65 34 76 78 97 46 32 51 21 comparisons = 22

15 inversions =

63 42 223 645 652 31 324 22 553 12 54 65 86 46 325 12 22 31 42 46 54 63 65 86 223 324 325 553 645 652

comparisons = 43

inversions =

II. Given an unsorted array of integers, design an algorithm and implement it using a program to sort an array of elements by partitioning the array into two subarrays based on a pivot element such that one of the sub array holds values smaller than the pivot element while another sub array holds values greater than the pivot element. Pivot element should be selected randomly from the array. Your program should also find number of comparisons and swaps required for sorting the array.

Input Format:

The first line contains number of test cases, T. For each test case, there will be two input lines. First line contains n (the size of array).
Second line contains space-separated integers describing array.

Output Format:
The output will have T number of lines.
For each test case T, there will be three output lines. First line will give the sorted array.
Second line will give total number of comparisons. Third line will give total number of swaps required.

Sample I/O Problem II:

Input: Output:

3 21 23 32 45 46 65 76 89

8 comparisons = 14

23 65 21 76 46 89 45 32 swaps = 10

10 21 32 34 46 51 54 65 76 78 97

54 65 34 76 78 97 46 32 51 21 comparisons = 29

	15 swaps = 21 63 42 223 645 652 31 324 22 553 12 54 65 86 46 325 12 22 31 42 46 54 63 65 86 223 324 325 553 645 652 comparisons = 45 swaps = 39 III. Given an unsorted array of integers, design an algorithm and implement it using a program to find Kth smallest or largest element in the array. (Worst case Time Complexity = O(n)) Input Format: The first line contains number of test cases, T. For each test case, there will be three input lines. First line contains n (the size of array). Second line contains space-separated integers describing array. Third line contains K. Output Format: The output will have T number of lines. For each test case, output will be the Kth smallest or largest array element. If no Kth element is present, output should be “not present”. Sample for Kth smallest: Input: Output: 3 123 10 78 123 656 54 765 344 514 765 34 765 234 3 15 43 64 13 78 864 346 786 456 21 19 8 434 76 270 601 8	
5.	Week 5: I. Given an unsorted array of alphabets containing duplicate elements. Design an algorithm and implement it using a program to find which alphabet has maximum number of occurrences and print it. (Time Complexity = O(n)) (Hint: Use counting sort) Input Format:	2

	The first line contains number of test cases, T. For each test case, there will be two input lines. First line contains n (the size of array). Second line contains space-separated integers describing array. Output: The output will have T number of lines. For each test case, output will be the array element which has maximum occurrences and its total number of occurrences. If no duplicates are present (i.e. all the elements occur only once), output should be “No Duplicates Present”. Sample I/O Problem I: Input: Output: 3 a – 3 10 No Duplicates Present a e d w a d q a f p l - 4 15 r k p g v y u m q a d j c z e 20 g t l l t c w a w g l c w d s a a v c l II. Given an unsorted array of integers, design an algorithm and implement it using a program to find whether two elements exist such that their sum is equal to the given key element. (Time Complexity = O(n log n)) Input Format: The first line contains number of test cases, T. For each test case, there will be two input lines. First line contains n (the size of array). Second line contains space-separated integers describing array. Third line contains key	

	Output Format: The output will have T number of lines. For each test case, output will be the elements arr[i] and arr[j] such that arr[i]+arr[j] = key if exist otherwise print 'No Such Elements Exist". Sample I/O Problem II: Input: Output: 2 10 40 10 No Such Element Exist 64 28 97 40 12 72 84 24 38 10 50 15 56 10 72 91 29 3 41 45 61 20 11 39 9 12 94 302 III. You have been given two sorted integer arrays of size m and n. Design an algorithm and implement it using a program to find list of elements which are common to both. (Time Complexity = O(m+n)) Input Format: First line contains m (the size of first array). Second line contains m space-separated integers describing first array. Third line contains n (the size of second array). Fourth line contains n space-separated integers describing second array. Output Format: Output will be the list of elements which are common to both. Sample I/O Problem III:	

	Input: Output: 7 10 10 34 55 34 76 10 39 85 10 55 12 30 55 34 72 10 34 10 89 11 30 69 51	
6.	Note: Consider the following input format in the form of adjacency matrix for graph based questions (directed/undirected/weighted/unweighted graph). Input Format: Consider example of below given graph in Figure (a). A boolean matrix AdjM of size V X V is defined to represent edges of the graph. Each edge of graph is represented by two vertices (start vertex u, end vertex v). That means, an edge from u to v is represented by making AdjM[u,v] and AdjM[v,u] = 1. If there is no edge between u and v then it is represented by making AdjM[u,v] = 0. Adjacency matrix representation of below given graph is shown in Figure (b). Hence edges are taken in the form of adjacency matrix from input. In case of weighted graph, an edge from u to v having weight w is represented by making AdjM[u,v] and AdjM[v,u] = w. Input format for this graph is shown in Figure (c). First input line will obtain number of vertices V present in graph. After first line, V input lines are obtained. For each line i in V, it contains V space separated boolean integers representing whether an edge is present between i and all V. Figure (a) Figure (b) Figure (c) Week 6: I. Given a (directed/undirected) graph, design an algorithm and implement it using a program to find if a path exists between two given vertices or not. (Hint: use DFS) Input Format: Input will be the graph in the form of adjacency matrix or adjacency list. Source vertex number and destination vertex number is also provided as an input. Output Format: Output will be 'Yes Path Exists' if path exists, otherwise print 'No Such	3

	Path Exists'. Sample I/O Problem I: II. Given a graph, design an algorithm and implement it using a program to find if a graph is bipartite or not. (Hint: use BFS) Input Format: Input will be the graph in the form of adjacency matrix or adjacency list. Output Format: Output will be 'Yes Bipartite' if graph is bipartite, otherwise print 'Not Bipartite'. Sample I/O Problem II: III. Given a directed graph, design an algorithm and implement it using a program to find whether cycle exists in the graph or not. Input Format: Input will be the graph in the form of adjacency matrix or adjacency list. Output Format: Output will be 'Yes Cycle Exists' if cycle exists otherwise print 'No Cycle Exists'. Sample I/O Problem III:	
7.	Week 7: Note: Input, output format along with sample input output for problem I and II is same and is provided at the end of problem II. I. After end term examination, Akshay wants to party with his friends. All his friends are living as paying guest and it has been decided to first gather at Akshay's house and then move towards party location. The problem is that no one knows the exact address of his house in the city. Akshay as a computer science wizard knows how to apply his theory subjects in his real life and came up with an amazing idea to help his friends. He draws a graph by looking in to location of his house and his friends' location (as a node in the graph) on a map. He wishes to find out shortest distance and path covering that distance from each of his friend's location to his house and then whatsapp them this path so that they can reach his house in minimum time. Akshay has developed the program that implements Dijkstra's algorithm but not	3

	sure about correctness of results. Can you also implement the same algorithm and verify the correctness of Akshay's results? (Hint: Print shortest path and distance from friends' location to Akshay's house) II. Design an algorithm and implement it using a program to solve previous question's problem using Bellman- Ford's shortest path algorithm. Input Format: Input will be the graph in the form of adjacency matrix or adjacency list. Source vertex number is also provided as an input. Output Format: Output will contain V lines. Each line will represent the whole path from destination vertex number to source vertex number along with minimum path weigth. Sample I/O Problem I and II: Input: Output: 5 1 : 0 0 4 1 0 0 2 3 1 : 3 0 0 0 0 4 3 1 : 1 0 2 0 4 0 4 3 1 : 3 0 0 0 0 4 5 2 3 1 : 7 0 0 0 0 0 1 III. Given a directed graph with two vertices (source and destination). Design an algorithm and implement it using a program to find the weight of the shortest path from source to destination with exactly k edges on the path. Input Format: First input line will obtain number of vertices V present in the graph. Graph in the form of adjacency matrix or adjacency list is taken as an input in next V lines. Next input line will obtain source and destination vertex number. Last input line will obtain value k. Output Format: Output will be the weigth of shortest path from source to destination having exactly k edges. If no path is available then print "no path of length k is available".	

	Sample I/O Problem III: Input: Output: 4 Weight of shortest path from (1,4) with 2 edges : 9 0 10 3 2 0 0 0 7 0 0 0 6 0 0 0 0 1 4 2	
8.	Week 8: Note: Input, output format along with sample input output for problem I and II is same and is provided at the end of problem II. I. Assume that a project of road construction to connect some cities is given to your friend. Map of these cities and roads which will connect them (after construction) is provided to him in the form of a graph. Certain amount of rupees is associated with construction of each road. Your friend has to calculate the minimum budget required for this project. The budget should be designed in such a way that the cost of connecting the cities should be minimum and number of roads required to connect all the cities should be minimum (if there are N cities then only N-1 roads need to be constructed). He asks you for help. Now, you have to help your friend by designing an algorithm which will find minimum cost required to connect these cities. (use Prim's algorithm) II. Implement the previous problem using Kruskal's algorithm. Input Format: The first line of input takes number of vertices in the graph. Input will be the graph in the form of adjacency matrix or adjacency list. Output Format: Output will be minimum spanning weight Sample I/O Problem I and II: Input: Output: 7 Minimum Spanning Weight: 39 0 0 7 5 0 0 0 0 0 8 5 0 0 0 7 8 0 9 7 0 0 5 0 9 0 15 6 0 0 5 7 15 0 8 9 0 0 0 6 8 0 11 0 0 0 0 9 11 0 III. Assume that same road construction project is given to another	3

	person. The amount he will earn from this project is directly proportional to the budget of the project. This person is greedy, so he decided to maximize the budget by constructing those roads who have highest construction cost. Design an algorithm and implement it using a program to find the maximum budget required for the project. Input Format: The first line of input takes number of vertices in the graph. Input will be the graph in the form of adjacency matrix or adjacency list. Output Format: Out will be maximum spanning weight. Sample I/O Problem III: Input: Output: 7 Maximum Spanning Weight: 59 0 0 7 5 0 0 0 0 0 8 5 0 0 0 7 8 0 9 7 0 0 5 0 9 0 15 6 0 0 5 7 15 0 8 9 0 0 0 6 8 0 11 0 0 0 0 9 11 0	
9.	Week 9: I. Given a graph, Design an algorithm and implement it using a program to implement Floyd- Warshall all pair shortest path algorithm. Input Format: The first line of input takes number of vertices in the graph. Input will be the graph in the form of adjacency matrix or adjacency list. If a direct edge is not present between any pair of vertex (u,v), then this entry is shown as AdjM[u,v] = INF. Output Format: Output will be shortest distance matrix in the form of V X V matrix, where each entry (u,v) represents shortest distance between vertex u and vertex v. Sample I/O Problem I: Input: Output: 5 Shortest Distance Matrix: 0 10 5 5 INF 0 10 15 5 15 INF 0 5 5 5 INF 0 5 5 5 INF INF 0 INF 10 INF INF 0 15 10 INF INF INF 0 20 INF INF INF 0 20 INF INF INF 5 0 INF INF INF 5 0	2

	II. Given a knapsack of maximum capacity w. N items are provided, each having its own value and weight. You have to Design an algorithm and implement it using a program to find the list of the selected items such that the final selected content has weight w and has maximum value. You can take fractions of items,i.e. the items can be broken into smaller pieces so that you have to carry only a fraction xi of item i, where 0 ≤xi≤ 1. Input Format: First input line will take number of items N which are provided. Second input line will contain N space-separated array containing weights of all N items. Third input line will contain N space-separated array containing values of all N items. Last line of the input will take the maximum capacity w of knapsack. Output Format: First output line will give maximum value that can be achieved. Next Line of output will give list of items selected along with their fraction of amount which has been taken. Sample I/O Problem II: Input: Output: 6 Maximum value : 22.33 6 10 3 5 1 3 item-weight 6 2 1 8 3 5 5-1 16 6-3 4-5 1-6 3-1 III. Given an array of elements. Assume arr[i] represents the size of file i. Write an algorithm and a program to merge all these files into single file with minimum computation. For given two files A and B with sizes m and n, computation cost of merging them is O(m+n). (Hint: use greedy approach) Input Format: First line will take the size n of the array. Second line will take array s an input. Output Format: Output will be the minimum computation cost required to merge all the elements of the array. Sample I/O Problem III:	

	Input: Output: 10 960 10 5 100 50 20 15 5 20 100 10 Solved example: Consider arr[5] = { 10, 5, 100, 50, 20, 15}. As per the brute force approach, first of all merge first two files (having 10 and 5 file size). Cost of merging will be = 10+5=15. List will become {15, 100, 50, 20, 15}. Similarly, again merging first two files (i.e. having 15 and 100 file size). Cost of merging will be = 15+100=115. List will become {115, 50, 20, 15}. For the subsequent steps the list becomes, (165, 20, 15}, {185, 15} and {200}. Therefore total cost of merging = 15+115+165+185+200 = 680. But this is not minimum computation cost. To find minimum cost, consider the order arr[5] = {5, 10, 15, 20, 50, 100}. By applying the same approach, the total cost of merging = 15+30+50+100+200 = 395.	
10.	Week 10: I. Given a list of activities with their starting time and finishing time. Your goal is to select maximum number of activities that can be performed by a single person such that selected activities must be non-conflicting. Any activity is said to be non-conflicting if starting time of an activity is greater than or equal to the finishing time of the other activity. Assume that a person can only work on a single activity at a time. Input Format: First line of input will take number of activities N. Second line will take N space-separated values defining starting time for all the N activities. Third line of input will take N space-separated values defining finishing time for all the N activities. Output Format: Output will be the number of non-conflicting activities and the list of selected activities. Sample I/O Problem I: Input: Output: 10 No. of non-conflicting activities: 4 1 3 0 5 3 5 8 8 2 12 List of selected activities: 1, 4, 7, 10 4 5 6 7 9 9 11 12 14 16 II. Given a long list of tasks. Each task takes specific time to accomplish it and each task has a deadline associated with it. You have to design an algorithm and implement it using a program to find maximum number of tasks that can be completed without crossing their	2

	deadlines and also find list of selected tasks. Input Format: First line will give total number of tasks n. Second line of input will give n space-separated elements of array representing time taken by each task. Third line of input will give n space-separated elements of array representing deadline associated with each task. Output Format: Output will be the total number of maximum tasks that can be completed. Sample I/O Problem II: Input: Output: 7 Max number of tasks = 4 2 1 3 2 2 2 1 Selected task numbers : 1, 2, 3, 6 2 3 8 6 2 5 3 III. Given an unsorted array of elements, design an algorithm and implement it using a program to find whether majority element exists or not. Also find median of the array. A majority element is an element that appears more than n/2 times, where n is the size of array. Input Format: First line of input will give size n of array. Second line of input will take n space-separated elements of array. Output Format: First line of output will be 'yes' if majority element exists, otherwise print 'no'. Second line of output will print median of the array. Sample I/O Problem III: Input: Output: 9 yes 4 4 2 3 2 2 3 2 2 2	
	Week 11: I.Given a sequence of matrices, write an algorithm to find most efficient way to multiply these matrices together. To find the optimal solution, you need to find the order in which these matrices should be multiplied. Input Format: First line of input will take number of matrices n that you need to multiply.	1

	For each line i in n, take two inputs which will represent dimensions aXb of matrix i. Output Format: Output will be the minimum number of operations that are required to multiply the list of matrices. Sample I/O Problem I: Input: Output: 3 4500 10 30 30 5 5 60 Solved Example: Consider a sequence of three matrices A of size 10X30, B of size 30X5, C of size 5X60. Then, (AB)C = (10*30*5) + (10*5*60) = 4500 operations A(BC) = (30*5*60) + (10*30*60) = 27000 operations. Hence the ouput of the program must be 4500 II.Given a set of available types of coins. Let suppose you have infinite supply of each type of coin. For a given value N, you have to Design an algorithm and implement it using a program to find number of ways in which these coins can be added to make sum value equals to N. Input Format: First line of input will take number of coins that are available. Second line of input will take the value of each coin. Third line of input will take the value N for which you need to find sum. Output Format: Output will be the number of ways. Sample I/O Problem II: Input: Output: 4 5 2 5 6 3 10 Solved Example: Let coin value set is C = {2,3,6,5} and the value N = 10. There are five solutions: {2,2,2,2,2}, {2,2,3,3}, {2,2,6}, {2,3,5} and {5,5}. Hence the output is 5. III.Given a set of elements, you have to partition the set into two subsets such that the sum of elements in both subsets is same. Design	

	an algorithm and implement it using a program to solve this problem. Input Format: First line of input will take number of elements n present in the set. Second line of input will take n space-separated elements of the set. Output Format: Output will be 'yes' if two such subsets found otherwise print 'no'. Sample I/O Problem III: Input: Output: 7 yes 1 5 4 11 5 14 10 Solved Example: Let set is S = {1, 5, 4, 11, 5, 14, 10}. Sum of the elements = 1+5+4+11+5+14+10 = 50. Now dividing the set into two halves such that sum of elements of both the subsets = (50/2) = 25. Therefore, subsets are {1, 5, 5, 14} and {4, 11, 10}.	
	Week 12: I.Given two sequences, Design an algorithm and implement it using a program to find the length of longest subsequence present in both of them. A subsequence is a sequence that appears in the same relative order, but not necessarily contiguous. Input Format: First input line will take character sequence 1. Second input line will take character sequence 2. Output Format: Output will be the longest common subsequence along with its length. Sample I/O Problem I: Solved Example: Consider two input strings “AGGTAB” and “GXTXAYB. Then the length of longest common subsequence is 4 i.e. for subsequence “GTAB”. II.Given a knapsack of maximum capacity w. N items are provided, each having its own value and weight. Design an algorithm and implement it using a program to find the list of the selected items such that the final selected content has weight <= w and has maximum value. Here, you cannot break an item i.e. either pick the complete item or don't pick it. (0-1 property). Input Format:	2

	First line of input will provide number of items n. Second line of input will take n space-separated integers describing weights for all items. Third line of input will take n space-separated integers describing value for each item. Last line of input will give the knapsack capacity. Output Format: Output will be maximum value that can be achieved and list of items selected along with their weight and value. Sample I/O Problem I: Input: Output: 5 Value = 16 2 3 3 4 6 Weights selected : 3 3 4 1 2 5 9 4 Values of selected weights : 2 5 9 10 III.Given a string of characters, design an algorithm and implement it using a program to print all possible permutations of the string in lexicographic order. Input Format: String of characters is provided as an input. Output Format: Output will be the list of all possible permutations in lexicographic order. Sample I/O Problem II: Input: Output: CAB ABC ACB BAC BCA CAB CBA	
	Week 13: I.Given an array of characters, you have to find distinct characters from this array. Design an algorithm and implement it using a program to solve this problem using hashing. (Time Complexity = O(n)) Input Format: First line of input will give the size n of the character array. Second line of input will give n space-separated elements to character array.	2

	Output Format: Output will be the list of characters present in the array in alphabetical order and frequency of each character in the array. Sample I/O Problem I: II.Given an array of integers of size n, design an algorithm and write a program to check whether this array contains duplicate within a small window of size k < n. Input Format: First input line contains number of test cases T. For each test case T, there will be three input lines. First line contains size n of array. Second input line contains n space-separated array elements. Third input line contains value k. Output Format: Output will have T number of lines. For each test case, output will be “Duplicate present in window k” if the duplicate element is found in the array, otherwise “Duplicate not present in window k”. Sample I/O Problem II: Input: Output: 2 Duplicate not present in window 3. 10 Duplicate present in window 4. 1 2 3 4 1 2 3 4 1 2 3 12 1 2 3 1 2 3 1 2 3 1 2 3 4 III.Given an array of nonnegative integers, Design an algorithm and implement it using a program to find two pairs (a,b) and (c,d) such that a*b = c*d, where a, b, c and d are distinct elements of array. Input Format: First line of input will give size of array n. Second line of input will give n space-separated array elements. Output Format: First line of output will give pair (a,b) Second line of output will give pair (c,d). Sample I/O Problem III: Input: Output: 10 4 2	

	31 23 4 1 39 2 20 27 8 10 1 8	
	Week 14: I.Given a number n, write an algorithm and a program to find nth ugly number. Ugly numbers are those numbers whose only prime factors are 2, 3 or 5. The sequence 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, 18, 20, 24,. is sequence of ugly numbers. Input: First line of input will give number of test cases T. For each test case T, enter a number n. Output: There will be T output lines. For each test case T, Output will be nth ugly number. Sample I/O Problem I: II.Given a directed graph, write an algorithm and a program to find mother vertex in a graph. A mother vertex is a vertex v such that there exists a path from v to all other vertices of the graph. Input: Graph in the form of adjacency matrix or adjacency list is provided as an input. Output: Output will be the mother vertex number. Solved Example: Consider a directed graph: In this graph, vertex 0 is mother vertex.	1
	Total	**24**

Text Books:

Authors Name	**Title**	**Edition**	**Publisher, Country**	**Year**
Thomas H. Cormen, Charles E. Leiserson, Ronal L. Rivest, and Clifford Stein	Introduction to Algorithms	4th Edition	MIT Press	2022

Reference Books:

Authors Name	Title	Edition	Publisher, Country	Year
Donald Knuth	Art of Computer Programming, The: Volume 1: Fundamental Algorithms (ART OF COMPUTER PROGRAMMING)	3rd Edition	Addison-Wesley	1998
Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran:"	Fundamentals of Computer Algorithms	2nd Edition	Universities press	2007
Anany Levitin	Introduction to the Design & Analysis of Algorithms	2nd Edition	Pearson Education	2008

Course Articulation Matrix

CO	Statement	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
PCS409.1	Analyse algorithmic time and space complexity using asymptotic notations.	-	3	-	1	-	-	-	-	1	1	-	1	2	2	2
PCS409.2	Design efficient algorithms utilizing techniques such as divide and conquer, greedy, and dynamic programming.	1	1	3	1	1	-	1	-	1	1	-	2	3	2	2
PCS409.3	Solve complex problems using backtracking and branch-and-bound techniques.	3	-	-	3	2	-	-	1	-	1	1	2	2	2	2
PCS409.4	Predict the complexity of NP-complete problems and propose algorithmic approaches.	1	-	1	3	1	-	-	1	-	1	-	1	2	3	2
PCS409.5	Apply Dijkstra's, Bellman-Ford, Prim's, and Kruskal's algorithms to real-world problems.	3	1	-	-	-	-	-	-	1	2	-	1	3	2	1
PCS409.6	Implement pattern matching algorithms like Rabin-Karp and brute-force techniques for pattern identification.	1	-	1	3	-	-	-	-	1	1	-	1	2	3	2
PCS 409		1.8	1.66	1.66	2.2	1.33	-	1	1	1	1.16	1	1.33	2.33	2.33	1.83

High correlation (3); Medium correlation (2); Low correlation (1), No correlation (-)

GRAPHIC ERA (DEEMED TO BE UNIVERSITY), DEHRADUN

SEMESTER IV

Name of Department: - Computer Science and Engineering

1. Subject Code: **TCS451** Course Title: **Virtualization and Cloud Computing**
2. Contact Hours: **L:** **3** **T**: **1** **P**: **0**
3. Examination Duration (Hrs): **Theory** **3** **Practical** **0**
4. Relative Weight: **CIE** **25** **MSE** **25** **ESE** **50**
5. Credits: **3**
6. Semester: **IV**
7. Category of Course: **DSE**
8. Pre-requisite: Fundamental of Computer & Introduction to Programming(TCS 101), Fundamental of Cloud Computing and Bigdata(TCS351)

9. Course Outcome:	After completion of the course the students will be able to**:** CO1 Discuss the different paradigms of cloud computing. CO2 Contrast parallel and distributed computing. CO3 Identify the concept of virtualization technique. CO4 Apply virtualization technique in cloud computing platform. CO5 Describe the architectures of cloud computing. CO6 Demonstrate the Use case of the virtualization and cloud computing services.

10. **Details of the Course:**

UNIT	CONTENTS	Contact Hrs
Unit - I	**Introduction to Cloud Computing** Why Cloud Computing (CC)? Different Perspectives on CC, Different Stakeholders in CC, Total cost of ownership (TCO) of on-premises IT, Cloud Computing Taxonomy, Characteristics of cloud computing, Characteristics of cloud computing as per NIST, Cloud Definitions. Cloud Computing at a Glance, The Vision of Cloud Computing, Cloud Computing Reference Model, Challenges Ahead, Historical Developments, Distributed Systems, Virtualization, Web 2.0, Service-Oriented Computing, Utility-Oriented Computing, Building Cloud Computing Environments, Application Development, Infrastructure and System Development, Computing Platforms and Technologies, Amazon Web Services (AWS), Google	9

	AppEngine, Microsoft Azure, Hadoop, Force.com and Salesforce.com	
Unit - II	**Virtualization** Introduction, Characteristics of Virtualized Environments, Taxonomy of Virtualization Techniques, Execution Virtualization, Types of hardware virtualization: Full virtualization - partial virtualization - para virtualization Desktop virtualization: Software virtualization – Memory virtualization - Storage virtualization – Data Virtualization – Network virtualization, Virtualization and Cloud Computing, Pros and Cons of Virtualization, Technology Examples, Xen: Para virtualization, VMware: Full Virtualization, Microsoft Hyper-V.	9
Unit – III	**Virtual Machines** Virtual machines basics, Process virtual machines: Memory architecture emulation, Instruction emulation, Operating system emulation, Dynamic binary optimization, High level VN architecture, System virtual machines: Resource virtualization (Processors, Memory, Input/Output), Case Study of Intel VT-x	8
Unit – IV	**Parallel and Distributed Computing** Eras of Computing, Parallel vs. Distributed Computing, Elements of Parallel Computing, What is Parallel Processing?, Hardware Architectures for Parallel Processing, Approaches to Parallel Programming, Levels of Parallelism, Laws of Caution, Elements of Distributed Computing, General Concepts and Definitions, Components of a Distributed System, Architectural Styles for Distributed Computing, Models for Inter-Process Communication, Technologies for Distributed Computing, Remote Procedure Call, Distributed Object Frameworks, Service Oriented Computing	8
Unit – V	**Cloud Computing Architecture** Fundamental Cloud Architectures - Workload Distribution Architecture - Resource Pooling Architecture - Dynamic Scalability Architecture – Elastic Resource Capacity Architecture -Service Load Balancing Architecture – Cloud Bursting Architecture - Elastic Disk Provisioning Architecture – Redundant Storage Architecture. Cloud Computing Reference Architecture (CCRA): Introduction, benefits of CCRA, Migrating into a Cloud: Introduction, Challenges while migrating to Cloud, Broad approaches to migrating into the cloud, Seven-step model of migration into a cloud, Migration Risks and Mitigation.	9
	Total	**43**

Text Books:

Authors Name	Title	Edition	Publisher, Country	Year
Rajkumar Buyya, Christian Vecchiola, S.Thamarai Selvi	Mastering Cloud Computing	1st Edition	McGraw Hill Education	2017
Jim Smith , Ravi Nair	Virtual Machines: Versatile Platforms for Systems and Processes	1st Edition	Morgan Kaufmann	2005
Pachghare V. K.	Cloud Computing	1st Edition	PHI Learning Pvt Ltd	2016

Reference Books:

Authors Name	Title	Edition	Publisher, Country	Year
Barrie Sosinsky	Cloud Computing Bible	1st Edition	Wiley	2011

Course Articulation Matrix

CO	Statement	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
TCS451.1	Discuss the different paradigms of cloud computing.															
TCS451.2	Contrast parallel and distributed computing.															
TCS451.3	Identify the concept of virtualization technique.															
TCS451.4	Apply virtualization technique in cloud computing platform.															
TCS451.5	Describe the architectures of cloud computing.															
TCS451.6	Demonstrate the Use case of the virtualization and cloud computing services.															
TCS 451																

High correlation (3); Medium correlation (2); Low correlation (1), No correlation (-)

GRAPHIC ERA (DEEMED TO BE UNIVERSITY), DEHRADUN

SEMESTER IV

Name of Department: - Computer Science and Engineering

1. Subject Code: **TCS471** Course Title: **Statistical Data Analysis with R**
2. Contact Hours: **L:** **3** **T**: **1** **P**: **0**
3. Examination Duration (Hrs): **Theory** **3** **Practical** **0**
4. Relative Weight: **CIE** **25** **MSE** **25** **ESE** **50**
5. Credits: **3**
6. Semester: **IV**
7. Category of Course: **DSE**
8. Pre-requisite: Engineering Mathematics-I (TMA101), Programming for problem solving(TCS 201), Fundamental of Cloud Computing and Bigdata(TCS351)

9. Course Outcome:	After completion of the course the students will be able to**:** CO1 Understand the concepts of statistics CO2 Apply the probability distribution techniques in different applications. CO3 Understand the needs of data pre-processing CO4 Implement the manipulation and processing of data in R CO5 Apply the concepts of functions in R CO6 Understand the use of R in data Analytics

10. **Details of the Course:**

UNIT	CONTENTS	Contact Hrs
Unit – I	**Statistics:** Introduction to Statistics- Descriptive Statistics, Summary Statistics Basic probability theory, Statistical Concepts (uni-variate and bi-variate sampling, distributions, re-sampling, statistical Inference, prediction error),	9
Unit - II	**Probability Distribution:** Introduction to Probability**,** Probability Distribution (Continuous and discrete- Normal, Bernoulli, Binomial, Negative Binomial, Geometric and Poisson distribution) , Bayes' Theorem, Central Limit theorem, Data Exploration & preparation, Concepts of Correlation,	10

	Regression, Covariance, Outliers.	
Unit – III	**Introduction to R and Data Pre-processing:** Introduction & Installation of R, R Basics, Finding Help, Code Editors for R, Command Packages, Manipulating and Processing Data in R, Reading and Getting Data into R, Exporting Data from R	10
Unit – IV	**Objects and Data Types**: Data Objects-Data Types & Data Structure. Viewing Named Objects, Structure of Data Items, Manipulating and Processing Data in R (Creating, Accessing, Sorting data frames, Extracting, Combining, Merging, reshaping data frames), Control Structures	8
Unit – V	**Functions:** Functions in R (numeric, character, statistical), working with objects, Viewing Objects within Objects, Constructing Data Objects, Building R Packages, Running and Manipulating Packages, Non parametric Tests- ANOVA, chi-Square, t-Test, U-Test, Introduction to Graphical Analysis, Using Plots(Box Plots, Scatter plot, Pie Charts, Bar charts, Line Chart), Plotting variables, Designing Special Plots, Simple Liner Regression, Multiple Regression	9
	Total	**46**

Text Books:

Authors Name	**Title**	**Edition**	**Publisher, Country**	**Year**
Mark Gardener	Beginning R: The Statistical Programming Language	1st Edition	Wiley	2013
Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani	An Introduction to Statistical Learning: with Applications in R	2nd Edition	Springer	2021

Reference Books:

Authors Name	Title	Edition	Publisher, Country	Year
N Das	Statistical Methods (Combined edition volume 1 & 2) Paperback – 1 July 2017	1st Edition	McGraw Hill Education	2017

Course Articulation Matrix

CO	Statement	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
TCS471.1	Understand the concepts of statistics	2	2						2							1
TCS471.2	Apply the probability distribution techniques in different applications.	2	3	2	1	1			1	1	2		1		2	2
TCS471.3	Understand the needs of data pre-processing	1			2	1			2		1		1			2
TCS471.4	Implement the manipulation and processing of data in R		2		2	3			2	1	2		1		2	1
TCS471.5	Apply the concepts of functions in R	1	2	1		1			1				2			
TCS471.6	Understand the use of R in data Analytics	1			2	2			2	2	3	1	1		2	2
TCS471		1.4	2.25	1.5	1.75	1.6	-	-	1.66-	1.33-	2	1	1.2	-	2	1.6

High correlation (3); Medium correlation (2); Low correlation (1), No correlation (-)

GRAPHIC ERA (DEEMED TO BE UNIVERSITY), DEHRADUN

SEMESTER IV

Name of Department: - Computer Science and Engineering

1. Subject Code: **TCS431** Course Title: **Microcontroller and Its Interfacing**
2. Contact Hours: **L:** 3 **T**: 1 **P**: 0
3. Examination Duration (Hrs): **Theory** 3 **Practical** 0
4. Relative Weight: **CIE** 25 **MSE** 25 **ESE** 50
5. Credits: 3
6. Semester: IV
7. Category of Course: DSE
8. Pre-requisite: Fundamental of IoT (TCS331)

9. Course Outcome:	After completion of the course the students will be able to**:** CO1 Understanding the concept of embedded system. CO2 Assembly language programming of 8051 CO3 Study of Arduino. CO4 Interfacing of different IC with 8051. CO5 Design and develop systems based on 8051 micro-controller and its interfaces. CO6 Understand the working of interrupts

10. **Details of the Course:**

Sl. No.	Contents	Contact Hours
1	**MICROCONTROLLER:** Difference between Microprocessors and Micro-controllers, Types of Micro-controllers, Memory structure of 8051, Processor Architecture – Harvard v/s Von Neumann, CISC v/s RISC, 8051 Architecture ,Micro-controller Memory types – control storage, variable area, stack, hardware register space, SFR,8051 pin diagram..	10
2	**8051 Instruction Set:** Addressing modes, external addressing, Instruction execution, Instruction set – data movement, arithmetic, bit operators,	9

	branch, Software development tools like assemblers, simulators, O/P file formats. Assembling and running an 8051 program, 8051 data types, 8051 flag bits and the PSW register, 8051 register banks and stack	
3	**PROGRAMMING OF 8051 and INTERRUPTS:** Programming of 8051, I/O bit manipulation. Timer, counter, programming of timer, 8051 interrupts, Interrupts priority in the 8051, and interrupts programming.	9
4	**INTRODUCTION TO ARDUINO IDE PLATFORM** Introduction to ATMEGA328 microcontroller and to Arduino IDE, Hardware, Characteristics, Interfacing with different peripheral devices, Debugging hardware errors, Using PWM I/O pins, Interfacing Arduino hardware with Internet of Things	9
5	**INTERFACING:** Interfacing with 8051: LCD, Keyboard, ADC, DAC interfacing, Sensor interfacing and Signal Conditioning, Stepper motor and DC motor, Basics of serial communications, 8051 connection to RS-232, 8051 serial port programming assembly.	8
	Total	**45**

Text Books:

Authors Name	**Title**	**Edition**	**Publisher, Country**	**Year**
Mazidi, Mazidi, and McKinay	The 8051 Microcontrollers & Embedded System	2nd Edition	Pearson Education India	2007
Myke Predko	Programming and Customizing the 8051 Microcontroller	1st Edition	McGraw Hill Education	2000
Brad Kendall	Getting Started With Arduino: A Beginner's Guide	-	-	2013

Reference Books:

Authors Name	Title	Edition	Publisher, Country	Year
Kenneth J. Ayala	The 8051 Microcontroller, 3rd Edition	3rd Edition	Cengage Learning	2004
Julien Bayle	C Programming for Arduino	-	Packt Pub Ltd	2013

Course Articulation Matrix

CO	Statement	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
TCS431.1	Understanding the concept of embedded system.															
TCS431.2	Assembly language programming of 8051															
TCS431.3	Study of Arduino															
TCS431.4	Interfacing of different IC with 8051.															
TCS431.5	Design and develop systems based on 8051 micro-controller and its interfaces.															
TCS431.6	Understand the working of interrupts															
TCS431																

High correlation (3); Medium correlation (2); Low correlation (1), No correlation (-)

GRAPHIC ERA (DEEMED TO BE UNIVERSITY), DEHRADUN

SEMESTER IV

Name of Department: - Computer Science and Engineering

1. Subject Code: **TCS 495** Course Title: **Foundation of Cyber Security**
2. Contact Hours: **L:** **4** **T**: **0** **P**: **0**
3. Examination Duration (Hrs): **Theory** **3** **Practical** **0**
4. Relative Weight: **CIE** **25** **MSE** **25** **ESE** **50**
5. Credits: **3**
6. Semester: **IV**
7. Category of Course: **DSE**
8. Pre-requisite: Introduction to Cryptography (TCS392)

9. Course Outcome:	After completion of the course the students will be able to**:** CO1 Explain different cyber threats and attacks CO2 Know the working of various cyber-attacks and cyber security protocols CO3 Analyse the different cyber security protocols. CO4 Use scripting language to implement security protocols. CO5 Apply security techniques to secure web applications CO6 Develop cyber security protocols.

10. **Details of the Course:**

Sl. No.	Contents	Contact Hours
1	**Unit 1:** Introduction to Cyber Security What is Cyber security, why we need Cyber security, The Zero Trust Model, Overview of ethical hacking. Protect Against - Unauthorized Modification, Unauthorized Deletion and Unauthorized Access. Three pillars of Cyber Security - Confidentiality, Availability and Integrity. Steps to fix a crime - Identify Cyber Threats, Analyse and Evaluate Threat, Treatment. Type of Hackers - White Hat, Great Hat, Black Hat. Penetration Testing and its Phases - Reconnaissance, Scanning, Gaining Access, Maintaining Access, Covering Tracks.	9
2	**Unit 2**: Linux Basics and Scripting for Ethical Hacking	10

	Bash, Linux commands, man page, adding and deleting, users and adding them to sudo group, switching users, creating, copying, moving and removing file, Writing and appending text to a file, file permissions, working with editors, grep, cut command, starting and stopping services Introduction to Bash scripting - Basics of Bash or Shell scripting, conditional statements, loops, manipulating files Introduction to Python - Basics of Python, conditional statements, loops, list, tuple, dictionary, functions.	
3	**Unit 3:** Networking Basics for Ethical Hacking Virtualization - Installing and configuring virtual machine, Network address translation, differences of IPv4 and IPv6, IP Address, Mac Address, TCP 3-way handshake, netcat - The Swiss Army Knife of TCP/IP Connections, use netcat to Listen on a port, pushing a command shell back to listener, transfer files, ICMP and Ping command, use of Wireshark tool.	9
4	**Unit 4:** Basics of Web and Web Security The client-server model for the web, various web threats and attacks, web cross site scripting (XSS) attack and use of scripting languages, phishing attacks, spear phishing, SQL injection attack, use of web penetration testing tools.	9
5	**Unit 5:** Introduction to Cyber Threats and System Hacking Cyber threats - malware, password attacks, distributed denial-of-service (DDos), ransomware attack, eavesdropping attack (man in the middle attack), birthday attack, IP and mac address spoofing, anonymous browsing and use of tor browser.	10
	Total	**47**

Text Books:

Authors Name	Title	Edition	Publisher, Country	Year
Anne Kohnke, Dan Shoemaker, Ken E. Sigler	The Complete Guide to Cybersecurity Risks and Controls (Internal Audit and IT Audit)	1st edition	Taylor & Francis Ltd	2022
	Penetration	1st Edition	No Starch Press,	2014

Georgia Weidman	Testing: A Hands-On Introduction to Hacking		USA	
Nina Godbole and Sunit Belapure	Cyber security : understanding cyber crimes, computer forensics and legal perspectives	1st Edition	Wiley, India	2011

Reference Books:

Authors Name	Title	Edition	Publisher, Country	Year
OccupyTheWeb	Linux Basics for Hackers: Getting Started with Networking, Scripting, and Security in Kali	Illustrated Edition	No Starch Press, USA	2018

Course Articulation Matrix

CO	Statement	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
TCS495.1	Explain different cyber threats and attacks	2	1	3	2	2	3	3	3	1	3	2	3	1	3	2
TCS495.2	Know the working of various cyber-attacks and cyber security protocols	1	2	3	2	2	3	2	3	2	2	2	3	2	3	2
TCS495.3	Analyse the different cyber security protocols.	3	3	3	3	3	3	3	3	2	2	3	3	3	3	3
TCS495.4	Use scripting language to implement security protocols.	1	2	3	3	3	2	3	3	2	2	2	3	2	3	3
TCS495.5	Apply security techniques to secure web applications	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
TCS495.6	Develop cyber security protocols	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
TCS495		2.16	2.33	3	2.66	2.66	2.83	2.83	3	2.16	2.5	2.5	3	2.33	3	2.66

High correlation (3); Medium correlation (2); Low correlation (1), No correlation (-)

GRAPHIC ERA (DEEMED TO BE UNIVERSITY), DEHRADUN

SEMESTER IV

Name of Department: - Computer Science and Engineering

1. Subject Code: **TCS421** Course Title: **Fundamental of Statistics and AI**
2. Contact Hours: **L:** **3** **T**: **0** **P**: **0**
3. Examination Duration (Hrs): **Theory** **4** **Practical** **0**
4. Relative Weight: **CIE** **25** **MSE** **25** **ESE** **50**
5. Credits: **3**
6. Semester: **4**
7. Category of Course: **DSE**
8. Pre-requisite: Engineering Mathematics-I (TMA101), Engineering Mathematics-II (TMA201), Python Programming for Computing (TCS341)

9. Course Outcome:	After completion of the course the students will be able to**:** CO1 Demonstrate knowledge of statistical and exploratory data analysis data analysis techniques utilized in decision making. CO2 Apply principles of Data Science to the analysis of business problems. CO3 To use Machine Learning Algorithms to solve real-world problems. CO4 To provide data science solution to business problems and visualization. CO5 To learn the basic concepts and techniques of AI and machine learning CO6 To explore the various mechanism of Knowledge and Reasoning used for building expert system

10. **Details of the Course:**

S. No.	Contents	Contact Hours
1	**Unit 1:** **Introduction to AI** Definition, Problem, State space representation. Intelligent Systems: Categorization of Intelligent System, Components of AI Program,	10

	Foundations of AI, Applications of AI, Current trends in AI, Intelligent Agents: Anatomy, structure, Types	
2	**Unit 2:** **Problem solving** Solving problem by Searching: Problem Solving Agent, Formulating Problems. Uninformed Search Methods: Breadth First Search (BFS), Depth First Search (DFS), Depth Limited Search, Depth First Iterative Deepening (DFID), Informed Search Methods: Greedy best first Search, A* Search, Memory bounded heuristic Search. Local Search Algorithms and Optimization Problems: Hill climbing search Simulated annealing, Local beam search.	9
3	**Unit 3:** **An Introduction to Data Science, Data Processing and Visualization** Definition, working, benefits and uses of Data Science, Data science vs. Business Intelligence, The data science process, Role of a Data Scientist. Data Processing and Visualization**:** **Data** Formatting, Exploratory Data Analysis, Filtering, and hierarchical indexing using Pandas. Data Visualization: Basic Visualization Tools, Specialized Visualization Tools, Seaborn Creating and Plotting Maps.	9
4	**Unit 4:** **Statistical Data Analysis & Inference** Populations and samples, Statistical modelling, probability distributions, fittings a model, Statistical methods for evaluation, Exploratory Data Analysis, Getting started with R, Manipulating and Processing data in R, working with function in R, working with descriptive Statistics, Working with graph plot in R.	9

5	**Unit 5:** **Statistical Applications** Basic Statistical operations, Linear Regression Analysis, Logistic and Exponential Regression, Time Series Analysis, Probability Distribution, ANOVA, Correlation and Covariance.	8
	Total	**45**

Text Books:

Authors Name	Title	Edition	Publisher, Country	Year
Tom M. Mitchell	Machine Learning	1st Edition	McGraw Hill Education	2017
K.G. Srinivasa, G.M. Siddesh, Chetan Shetty, Sowmya B.J.	Statistical Programming in R	1st Edition	Oxford University Press	2017

Reference Books:

Authors Name	Title	Edition	Publisher, Country	Year

Course Articulation Matrix

CO	Statement	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
TCS421.1	Demonstrate knowledge of statistical and exploratory data analysis data analysis techniques utilized in decision making.	3	1	1			1						2		2	1
TCS421.2	Apply principles of Data Science to the analysis of business problems.	2	3	2	1	1			1	1	2		1		2	2
TCS421.3	To use Machine Learning Algorithms to solve real-world problems.	2	1		2	1			1		1		2		1	2
TCS421.4	To provide data science solution to business problems and visualization.		2		2	3			2	1	2		1		2	1
TCS421.5	To learn the basic concepts and techniques of AI and machine learning	1	2	1		1			1				2			
TCS421.6	To explore the various mechanism of Knowledge and Reasoning used for building expert system	1			2	2			2	2	3	1	1		2	2
TCS421		1.8	1.8	1.33	1.75	1.6	1	-	1.4	1.33	2	1	1.5	-	1.8	1.6

High correlation (3); Medium correlation (2); Low correlation (1), No correlation (-)