# GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

## SEMESTER IV

Name of Department: - Computer Science and Engineering

**Programming in Java**

* 1. Subject Code: Course Title:

**0**

**0**

**TCS-408**

* 1. Contact Hours: **L: T**: **P**:

**50**

**3**

**0**

* 1. Examination Duration (Hrs): **Theory 3**
  2. Relative Weight: **CIE. MSE**

**25**

**25**

* 1. Credits:

**DC**

**IV**

**3**

* 1. Semester:
  2. Category of Course:

## Practical SEE

* 1. Pre-requisite: TCS 307 Object oriented Programming with C++

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 applications.

CO6. Design applications for event handling and accessing databases using Java features.

9. **Course Outcome\*\*:**

*\*\* Describe the specific knowledge, skills or competencies the students are expected*

*to acquire or demonstrate.*

## Details of the Course:

|  |  |  |
| --- | --- | --- |
| **Sl.**  **No.** | **Contents** | **Contact**  **Hours** |
| 1 | **Unit 1:**  **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 |
| 2 | **Unit 2:**  **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. | 9 |

|  |  |  |
| --- | --- | --- |
|  | **Interface:** Defining Interfaces, abstract methods declarations, implementing interfaces, extended interfaces, interface references. |  |
| 3 | **Unit 3: Exception handling:** Exception Types, Exception class, RuntimeException Class, Error Class, Checked and uncheced 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 |
| 4 | **Unit 4: Collection and Generic Framework:** Introduction to Collection and Generic Framework: Object class and typesafety, Generics, Interfaces Iterator, List, Set, ArrayList, LinkedList HashSet and ArrayDeque 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 |
| 5 | **Unit 5: 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** |

1. **Suggested Books:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SL.**  **No.** | **Name of Authors/Books/Publishers** | **Edition** | **Year of**  **Publication / Reprint** |
|  | **Textbooks** |  |  |
| 1. | Patrick Naughton and Herbert Schildt, “Java 2 The Complete Reference”, , McGraw Hill Education | 9th | **2017** |
| 2 | E. Balaguruswamy, “Programming with Java a Primer”,  Tata McGraw Hill | 6th | **2019** |
|  | **Reference Books** |  |  |
| 1. | Core Java-Head First Publication | 3rd | 2018 |
| 2. | Cay S Horstmann and Gary Cornell, “Core Java  Volume –I and II”, Sun Microsystems, |  | **2019** |

|  |  |  |
| --- | --- | --- |
| 12. | **Mode of**  **Evaluation** | Test / Quiz / Assignment / Mid Term Exam / End Term Exam |

# GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

## SEMESTER IV

Name of Department: - Computer Science and Engineering

**Finite Automata and Formal Languages**

1. Subject Code: Course Title:

**TCS-402**

1. Contact Hours**: L:**

**3**

1. Examination Duration (Hrs):
2. Relative Weight: **CIE**

**25**

1. Credits:

**IV**

**4**

1. Semester:
2. Category of Course:

**DC**

**T**:

## Theory MSE

**25**

**1**

**P**:

## Practical SEE

**50**

**0**

**3**

**0**

1. Pre-requisite: TMA 101 / 201 Engineering Mathematics

CO1. Understand the concept of abstract machines and their power to recognize languages.

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 and be familiar with Turing machines and computability.

CO6: Formulate finite machines, push down automata and Turing machines for automated functioning of devices.

9. **Course Outcome\*\*:**

*\*\* Describe the specific knowledge, skills or competencies the students are expected*

*to acquire or demonstrate.*

1. Detailed Syllabus

|  |  |  |
| --- | --- | --- |
| **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, Myhill-Nerode Theorem. | **10** |
| **Unit - II** | Regular expression (RE), Definition, Operators of regular expression and their precedence, Algebraic laws for Regular expressions, 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 | **10** |

|  |  |  |
| --- | --- | --- |
|  | 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. |  |
| **Unit – III** | Context free grammar (CFG) and Context Free Languages (CFL): Definition, Examples, Derivation, Derivation trees, Ambiguity in Grammar, Useless symbols, Simplification of CFGs, Normal forms for CFGs: CNF and GNF, Closure proper ties of CFLs, Decision Properties of CFLs, Pumping lemma for CFLs, Applications and Limitation of CFL and CFG. | **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** |

1. **Suggested Books:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SL.**  **No.** | **Name of Authors/Books/Publishers** | **Edition** | **Year of Publication /**  **Reprint** |
|  | **Textbooks** |  |  |
| 1. | J. Hopcroft, R. Motwani, and J. Ullman. Introduction to  Automata Theory, Languages, and Computation, 3rd edition, 2007, Pearson/Addison-Wesley. | 3rd | 2007 |
| 2 | KLP Mishra and N. Chandrasekaran, “Theory of Computer Science: Automata, Languages and Computation”, PHI Learning Private Limited, Delhi  India. | 3rd | 2006 |
|  | **Reference Books** |  |  |
| 1. | Michael Sipser, Introduction to the Theory of  Computation, 3rd edition (or 1st edition), 2013, Cengage Learning | 1st | 2013 |
| 2. | P. Linz. Introduction to Formal Languages and  Automata, 6th edition, 2017, Jones and Barlett. | 6th | 2017 |

|  |  |  |
| --- | --- | --- |
| 12. | **Mode of**  **Evaluation** | Test / Quiz / Assignment / Mid Term Exam / End Term Exam |

# GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

## SEMESTER IV

Name of Department: - Computer Science and Engineering

**Microprocessors**

1. Subject Code: Course Title:

**TCS-403**

1. Contact Hours: **L:**

**P**:

**T**:

**0**

**0**

**3**

1. Examination Duration (Hrs): **Theory Practical**

**3**

**0**

1. Relative Weight: **CIE MSE SEE**

**50**

**25**

**25**

1. Credits:

**IV**

**3**

1. Semester:
2. Category of Course:

**DC**

1. Pre-requisite: TEC-101 Basic Electronics, TCS-301-Logic design

After completion of the course the students will be able to**:**

**CO1**: Remember the concept of microcomputer system.

**CO2**: Understand microprocessor 8085, 8086 and microcontroller 8051 hardware. **CO3**: Apply the concepts of assembly language programming of 8085 and 8086 to fulfil different tasks.

**CO4:** Examine the application of 8085 and 8086 microprocessor with interrupt system, real time timer and counter.

**CO5:** Test different interfacing ICs and memory for defined tasks with 8085 and 8086 microprocessor.

**CO6:** Integrate the knowledge of 8085, 8086and 8051 in various embedded systems.

9. **Course Outcome\*\*:**

*\*\* Describe the specific knowledge, skills or competencies the students are expected to*

*acquire or demonstrate.*

## Details of the Course:

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Contents** | **Contact**  **Hours** |
| 1 | **Unit1: Microprocessor Architecture:** Introduction to Microprocessor and Microcomputer Architecture, Brief Evolution of Microprocessor, 8085  Register Organization, Internal architecture, Pins & Signals, 8085 Interrupts. | 10 |
| 2 | **Unit 2: Programming with 8085:** Programming model of 8085, 8085 instructions set, Addressing modes, Instruction Timing & Execution, Assembly language Programs using Instruction of 8085. Stack and subroutine. | 10 |
| 3 | **Unit 3: Interfacing with 8085, Memory Interfacing**: Interfacing  EPROM&RAM memories. Address Decoding, Memory & I/O mapped I/O, | 10 |

|  |  |  |
| --- | --- | --- |
|  | Programmable Interfaces: - 8255, 8251, 8253, 8259, Overview of DMA &  DMA controller, ADC and DAC interfacing. |  |
| 4 | **Unit 4: Intel 8086 (16 bit processor):** 8086 Architecture, Physical address, segmentation, memory organization, Instruction set of 8086, Addressing Modes, Pins &Signals, 8086 Interrupts. Simple Assembly Language Programs  using Instruction of 8086. | 10 |
| 5 | **Unit 5: 8051 Microcontroller:** Microprocessor Vs Microcontroller,  Embedded Systems, 8051 Architecture- Registers, Pin diagram, I/O ports functions, Memory organization. Interrupts of 8051. | 08 |
|  | Total | **48** |

1. **Suggested Books:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SL.**  **No.** | **Name of Authors/Books/Publishers** | **Edition** | **Year of Publication /**  **Reprint** |
|  |  |  |  |
|  | **Textbooks** |  |  |
| 1. | Ramesh Gaonkar, Microprocessor Architecture, Programming, and Applications with the 8085, Penram  International Publication (India) Pvt. Ltd | 6th | 2013 |
| 2 | A. K. Ray & K. M. Bhurchandi, Advanced  Microprocessors and peripherals, Tata McGraw Hill | 3rd | 2012 |
| 3. | Muhammad Ali Mazidi, Janice Gillispie Mazidi, The 8051 Microcontroller & Embedded System, Pearson /  PHI publication | 2nd | 2007 |
|  | **Reference Books** |  |  |
| 1. | Douglas V. Hall, Microprocessors and Interfacing, Tata  McGraw Hill | 3rd | 2012 |
| 2. | Barry B. Brey, The Intel Microprocessors Architecture  Programming and interfacing, Pearson | 8th | 2012 |

|  |  |  |
| --- | --- | --- |
| 12. | **Mode of Evaluation** | Test / Quiz / Assignment / Mid Term Exam / End Term Exam |

# GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

## SEMESTER IV

Name of Department: - Computer Science and Engineering

**Design and Analysis of Algorithms**

* 1. Subject Code: Course Title:

**TCS 409**

* 1. Contact Hours: **L: T**: **P**:

**3**

**0**

**0**

* 1. Examination Duration (Hrs): **Theory Practical**

**0**

**3**

* 1. Relative Weight: **CIE MSE SEE**

**50**

**25**

**25**

* 1. Credits:

**IV**

**3**

* 1. Semester:
  2. Category of Course:

**DC**

* 1. Pre-requisite: TCS-302 Data Structures with C, Any programming language

After completion of the course the students will be able to**:**

CO1: Understand various asymptotic notations to analyze time and space complexity of algorithms.

CO2: Analyze 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.

9. **Course Outcome\*\*:**

*\*\* Describe the specific knowledge, skills or competencies the students are expected to*

*acquire or demonstrate.*

## Details of the Course:

|  |  |  |
| --- | --- | --- |
| **SL.**  **NO.** | **Contents** | **Contact**  **Hours** |
| 1 | **Unit 1:**  **Asymptotic Notations and Searching Algorithms**  **Introduction to Algorithms -** What is an Algorithm, Rate of growth, Commonly  used rate of growths, Types of analysis, Asymptotic Notations, Master theorem | **8** |

|  |  |  |
| --- | --- | --- |
|  | **Searching -** Linear search (sorted and unsorted), Iterative and recursive binary search, Tower of Hanoi and solving its recursion, Fibonacci and solving its  recursion |  |
| 2 | **Unit 2:**  **Sorting Algorithms Sorting -** Bubble sort, Insertion sort, selection sort, quick sort, randomized quick sort, merge sort, heap sort, counting sort, External sorting  **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** |
| 3 | **Unit 3:**  **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 **All pair shortest paths -** The  Floyd Warshall algorithm | **12** |
| **4** | **Unit 4:**  **Algorithm Design Techniques - Greedy and Dynamic Programming**  **Greedy algorithms -** Activity selection problem, Job sequencing problem, Huffman codes, fractional knapsack problem  **Dynamic Programming -** Overlapping substructure property, Optimal  substructure property, Tabulation vs Memoization, Fibonacci numbers, 0/1 Knapsack problem, Longest common subsequence, Matrix chain multiplication | **10** |
| **5** | **Unit 5:**  **Hashing, String Matching and NP-Completeness**  **Hashing Data Structure -** Introduction to Hashing, Hash function, Collision and collision handling, Collision handling - Chaining, Open addressing  **String Matching -** Naive string-matching algorithm, The Rabin-Karp algorithm, The Knuth-Morris-Pratt algorithm  **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** |

1. **Suggested Books:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SL.**  **No.** | **Name of Authors/Books/Publishers** | **Edition** | **Year of Publication /**  **Reprint** |
|  | **Textbooks** |  |  |
| 1. | Thomas H. Cormen, Charles E. Leiserson, Ronal L. Rivest, Clifford Stein:” Introduction to Algorithms”, 2nd Edition, PHI, | 2nd | 2006. |
|  | **Reference Books** |  |  |
| 1. | Donald E.Knuth:”The Art of Computer Programming: Volume  1: Fundamental Algorithms”,3rd Edition | 3rd | 1998 |

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | Ellis Horowitz, Sartaj Sahni, SanguthevarRajasekaran:” Fundamentals of Computer Algorithms”, 2nd Edition,  University press,. | 2nd | 2007 |
| 3 | Anany Levitin: “Introduction to the Design & Analysis of  Algorithms”, 2nd Edition, Pearson Education,. | 2nd | 2007 |

|  |  |  |
| --- | --- | --- |
| 12. | **Mode of Evaluation** | Test / Quiz / Assignment / Mid Term Exam / End Term Exam |

# GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

## SEMESTER IV

Name of Department: - Computer Science and Engineering

**0**

**Virtualization and Cloud Computing**

* 1. Subject Code: Course Title:

**TCS 451**

* 1. Contact Hours: **L: T**: **P**:

**0**

**3**

* 1. Examination Duration (Hrs): **Theory Practical**

**0**

**3**

* 1. Relative Weight: **CIE MSE SEE**

**50**

**25**

**25**

* 1. Credits:

**DE**

**IV**

**3**

* 1. Semester:
  2. Category of Cours

e:

* 1. Pre-requisite: TCS-351 Fundamental of Cloud Computing and Big data

After completion of the course the students will be able to**:**

CO1: Identify the concept of virtualization technique in cloud computing platform.

CO2: Demonstrate the use case of the virtual machines.

CO3: Analyze the use case of parallel and distributed computing.

CO4: Evaluate the architectures of cloud computing.

CO5: Assess the viability of developing, deploying, maintaining and securing cloud computing solutions using a variety of resiliency virtualization testing tools.

CO6:Create cloud-computing virtualization strategies using virtualization tools to solve identified business needs on behalf of a client.

9. **Course Outcome\*\*:**

*\*\* Describe the specific knowledge, skills or competencies the students are expected to*

*acquire or demonstrate.*

## Details of the Course:

|  |  |  |
| --- | --- | --- |
| **SL.**  **NO** | **Contents** | **Contact**  **Hours** |
| 1 | **Unit 1:**  **Virtualization**  Introduction, Characteristics of Virtualized Environments, Taxonomy of Virtualization Techniques, Execution Virtualization, Types of hardware  virtualization: Full virtualization - partial virtualization - para virtualization | **9** |

|  |  |  |
| --- | --- | --- |
|  | Application Virtualization, Network Virtualization, Desktop Virtualization, Storage Virtualization, Server Virtualization, Data Virtualization.  Virtualization Solution Providers: V2 Cloud, VMware, Citrix, Microsoft, Oracle VM VirtualBox.  Advantages and Disadvantages of Virtualization, Examples, Xen: Para  virtualization, VMware: Full Virtualization, Microsoft Hyper-V, Case study of Virtualization Technique |  |
| 2 | **Unit 2:**  **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 | **9** |
| 3 | **Unit 3:**  **Parallel and Distributed Computing**  Eras of Computing, Parallel vs. Distributed Computing, Architectural Styles for Distributed Computing, Models for Inter-Process Communication, Technologies for Distributed Computing, Remote Procedure Call, Distributed Object Frameworks, Service Oriented Computing, Parallel programming, from laptops to supercomputers to the cloud, Parallel architectures and programming languages and models, including shared memory (eg OpenMP on your multicore laptop), distributed memory (MPI and UPC on a supercomputer), GPUs (CUDA and OpenCL), and cloud (MapReduce, Hadoop and Spark). Parallel algorithms and software tools for common computations (eg dense and sparse linear algebra, graphs, structured grids). Tools for load balancing, performance analysis, debugging. How high level applications are built (eg climate modeling). Case study of Parallel and Distributed Computing. | **9** |
| 4 | **Unit 4:**  **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. Case study of Cloud Computing Architecture. | **9** |
| **5** | **Unit 5:**  **Virtualization Tools and Techniques**  Parasoft Virtualize, Mountebank, Hoverfly cloud, MicroFocus Data Simulation Software, CA service Virtualization, Mocklab, Rational Test Virtualization Server, Tricentis Tosca, Docker, Kubernetes, OpenShift, and Istio. Assess the viability of developing, deploying, maintaining and securing cloud computing solutions using a variety of resiliency  virtualization testing tools, Create cloud-computing virtualization | **9** |

|  |  |  |
| --- | --- | --- |
|  | strategies using virtualization tools to solve identified business needs on behalf of a client, Case study of Virtualization Tools and Techniques.  Design and Deploy an Online Healthcare Application on the Cloud. |  |
|  | **Total** | **45 Hrs.** |

1. **Suggested Books:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SL.**  **No.** | **Name of Authors/Books/Publishers** | **Edition** | **Year of Publica tion /**  **Reprint** |
|  | **Textbooks** |  |  |
| 1. | Mastering Cloud Computing by Rajkumar Buyya etc., Published by McGraw Hill, | 1st | 2017 |
| 2 | Virtual Machines by James E. Smith, Ravi Nair, Published by MK Publishers, | 1st | 2005 |
| 3 | V K Pachghare, Cloud Computing, PHI, |  | 2016 |
|  | **Reference Books** |  |  |
| 1. | Barrie Sosinsky , Cloud Computing Bible, Wiley Publishing Inc., | 1st | 2011 |

|  |  |  |
| --- | --- | --- |
| 12. | **Mode of Evaluation** | Test / Quiz / Assignment / Mid Term Exam / End Term Exam |

# GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

## SEMESTER IV

Name of Department: - Computer Science and Engineering

**0**

**Statistical Data Analysis with R**

* 1. Subject Code: Course Title:

**TCS 471**

* 1. Contact Hours: **L: T**: **P**:

**3**

**0**

* 1. Examination Duration (Hrs): **Theory Practical**

**0**

**3**

* 1. Relative Weight: **CIE MSE SEE**

**50**

**25**

**25**

* 1. Credits:

**3**

* 1. Semester:

**IV**

* 1. Category of Course:

**DE**

* 1. Pre-requisite: TMA101-Engineering Mathematics, TCS-201 Programming for problem solving, TCS 351 Fundamental of Cloud Computing and Big data

After completion of the course the students will be able to**:**

CO1: Define the concepts of statistics.

CO2: Understand the probability distribution techniques in different applications. CO3: Identify the needs of data preprocessing.

CO4: Implement the manipulation and processing of data in R CO5: Apply the concepts of functions in R

CO6: Evaluate the use of R in data analytics

9. **Course Outcome\*\*:**

*\*\* Describe the specific knowledge, skills or competencies the students are expected to*

*acquire or demonstrate.*

## Details of the Course:

|  |  |  |
| --- | --- | --- |
| **Sl.**  **No.** | **Contents** | **Contact**  **Hours** |
| 1 | **Unit 1:** 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 |
| 2 | **Unit 2: 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, Regression, Covariance, Outliers. | 10 |

|  |  |  |
| --- | --- | --- |
| 3 | **Unit 3:** Introduction to R and Data Preprocessing: 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 |
| 4 | **Unit 4:** 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 |
| 5 | **Unit 5:** 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 | 10 |
|  | Total | **47** |

1. **Suggested Books:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SL.**  **No.** | **Name of Authors/Books/Publishers** | **Edition** | **Year of**  **Publication / Reprint** |
|  | **Textbooks** |  |  |
| 1. | Dr. Mark Gardener, “Beginning R: The Statistical Programming Language”, John willey& Sons, 2012 | 1st | 2012 |
| 2. | Gareth James, Daniela Witten, Trevor Hastie and Robert  Tibshirani, “An Introduction to Statistical Learning with Applications in R,” Springer, Germany, | 2nd | 2021 |
|  | **Reference Books** |  |  |
| 1. | N.G Das, “Statistical Methods (Combined edition  volume 1 & 2),” Mc Graw Hill | 1st | 2017 |

|  |  |  |
| --- | --- | --- |
| 12. | **Mode of**  **Evaluation** | Test / Quiz / Assignment / Mid Term Exam / End Term Exam |

# GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

## SEMESTER IV

Name of Department: - Computer Science and Engineering

**Fundamental of Cyber Security**

* 1. Subject Code: Course Title:

**TCS 492**

* 1. Contact Hours: **L T**: **P**:

**3**

**0**

**0**

* 1. Examination Duration (Hrs): **Theory Practical**

**0**

**3**

* 1. Relative Weight: **CIE MSE SEE**

**50**

**25**

**25**

* 1. Credits:

**DE**

**IV**

**3**

* 1. Semester:
  2. Category of Course:
  3. Pre-requisite: TCS-332 Fundamentals of Information Security and Block Chain, TCS-392 Introduction to Cryptography

After completion of the course the students will be able to**:**

CO1: Explain the three pillars of cyber security, types of hackers and penetration testing.

CO2: . Implement the scripting concepts used in cyber security.

CO3: Use the netcat, ping and wireshark tools to analyze the security of network. CO4: Use the Javascript, php, sql to analyze the web security.

CO5: Explain the use of cyber security protocols for cyber threats. CO6: Analyze the security level of web applications.

9. **Course Outcome\*\*:**

*\*\* Describe the specific knowledge, skills or competencies the students are expected*

*to acquire or demonstrate.*

## 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, Ethical Hacking  Protect Against - Unauthorised Modification, Unauthorised Deletion and Unauthorised 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** |
|  | **Unit 2:**  **Linux Basics and Scripting for Ethical Hacking** | **9** |

|  |  |  |
| --- | --- | --- |
| 2 | 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, Automating tasks with cron jobs **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, Configuration of network - NAT, Bridged and Host only, Dual boot system  **TCP/IP -** IPv4 and IPv6, IP Address, Mac Address, Subnets, TCP 3-way handshake, DNS  Communication between VMs or Setting up network between machines,  **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, Ping command **Wireshark -** A Packet Sniffing Tool | **10** |
| **4** | **Unit 4:**  **Basics of Web and Web Security**  **Introduction to Java Script -** Basics of Javascript, Input validation, Cross site scripting (XSS) attack  **Introduction to PHP and SQL -** Basics of PHP, Input Validation, Phishing, Spear Phishing, Sql Injection Attack | **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, buffer overflow attack, IP and Mac address spoofing, Steganography, Anonymous browsing, Introduction to tor browser, Introduction  to VPN, Secure Sockets Layer (SSL), Secure Shell (SSH) | **10** |
|  | **Total** | **47** |

1. **Suggested Books:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SL.**  **No.** | **Name of Authors/Books/Publishers** | **Edition** | **Year of Publication /**  **Reprint** |
|  | **Textbooks** |  |  |
| 1. | Georgia Weidman ,“Penetration Testing: A Hands-On Introduction to Hacking,” No Starch Press Inc., | 1st | 2014. |
| 2 | Nina Godbole ,SunitBelapure, “Cyber Security:Understanding Cyber Crimes, Cyber Forensics  and Legal Perspectives,” Wiley India,. | 1st | 2011 |

# GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

## SEMESTER IV

Name of Department: - Computer Science and Engineering

**Fundamental of Statistics and AI**

* 1. Subject Code: Course Title:

**TCS 421**

* 1. Contact Hours: **L: T**: **P**:

**50**

**0**

**0**

**0**

**3**

* 1. Examination Duration (Hrs): **Theory 3**

**25**

**25**

* 1. Relative Weight: **CIE MSE**
  2. Credits:

**IV**

**3**

* 1. Semester:
  2. Category of Course:

**DE**

## Practical SEE

* 1. Pre-requisite: TMA101/TMA Engineering Mathematics, TCS-341 Python Programming for Computing

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

9. **Course Outcome\*\*:**

*\*\* Describe the specific knowledge, skills or competencies the students are expected to*

*acquire or demonstrate.*

## Details of the Course:

|  |  |  |
| --- | --- | --- |
| **SL.**  **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, Foundations of AI, Applications of AI, Current trends in AI, Intelligent Agents: Anatomy, structure, Types | 10 |

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

1. **Suggested Books:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SL.**  **No.** | **Name of Authors/Books/Publishers** | | **Edition** | **Year of**  **Publication / Reprint** |
|  | | **Textbooks** |  |  |
| 1. | | Tom M. Mitchell. "Machine Learning" McGraw-Hill,. | 1st | 2017 |
| 2 | | “Statistical programming in R”, Oxford University Press | 1st | 2017 |

|  |  |  |
| --- | --- | --- |
| 12. | **Mode of Evaluation** | Test / Quiz / Assignment / Mid Term Exam / End Term Exam |

# GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

## SEMESTER IV

Name of Department: - Computer Science and Engineering

**Java Programming Lab**

* 1. Subject Code: Course Title:

**PCS-408**

* 1. Contact Hours: **L: T**: **P**:

**1**

**2**

**0**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 3. Examination Duration (Hrs): |  | **Theory** | **0** |  | **Practical** |  | **3** |
|  |  |  |  |  |  |  |  |
| 4. Relative Weight: **CIE 25** |  | **MSE** |  | **25** | **SEE** |  | **50** |
|  |  |  |  |  |  |  |  |
| 5. Credits: **2** | |  | | | | | |
| 6. Semester: **IV** |  |
| 7. Category of Course: **DC** | |

8. Pre-requisite: PCS-307 OOPS with C++ Lab, TCS 307 Object oriented Programming with C++

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: Analyze, 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.

9. **Course Outcome\*\*:**

*\*\* Describe the specific knowledge, skills or competencies the students are expected to acquire*

*or demonstrate.*

## Details of the Course:

|  |  |  |
| --- | --- | --- |
| **Sl.**  **No.** | **List of problems for which student should develop program and execute**  **in the Laboratory** | **Contact**  **Hours** |
| 22 | **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, universityrollno and semester. Display the information.  Name: UniversityRollNo:Course: | 2 |

|  |  |  |
| --- | --- | --- |
|  | Semester: |  |
| 23 | **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:   1. Term Deposit 2. 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 installment (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 |
| 24 | Program to find if the given numbers are Friendly pair or not (Amicable or not). **Friendly Pair** are two or morenumbers 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".   + Else, print "Not Abundant Numbers".   Program to check whether the given numbers are friendly pair or not | 2 |
| 25 | 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 | 2 |

|  |  |  |
| --- | --- | --- |
|  | * Input the integer from the user. * Traverse the integer digit by digit. * If a '0' is encountered, replace it by '1'. * Print the integer. |  |
| 26 | **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 |
| 27 | The problem to rearrange positive and negative numbers in an [array](https://www.faceprep.in/procoder/knowledgebase/introduction-to-arrays) .  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 | 2 |
| 28 | 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. | 2 |

|  |  |  |
| --- | --- | --- |
|  | 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. |  |
| 29 | **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.  **Input:** 01101111010  **Output:** 3  **Explanation: 0110**1111010 - count = 1 | 2 |
| 30 | Write a java program to delete vowels from given string using StringBuffer class | 2 |
| 31 | **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 | 2 |

|  |  |  |
| --- | --- | --- |
|  | **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.   1. Change the address of any depositor and then display the final information of that depositor. 2. Randomly repeat these processes for some other   bank accounts. |  |
| 32 | 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 |
| 33 | **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)** |  |
| 34 | **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 |
| 35 | 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:**  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 | 2 |
| 36 | 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 | 2 |

|  |  |  |
| --- | --- | --- |
|  | 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 |  |
| 37 | 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:   1. First Letter of employee name should be in capital letter. 2. Employee id should be between 2001 and 5001 3. 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 |
|  | 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". |  |
|  | **Sample Input** |  |
|  | 3 5 |  |
| 38 | 2 4 | 2 |
|  | 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. |  |
| 39 | **File Handling in Java:**  Write a java file handling program to count and display the 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 |
| 40 | **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. |  |
| 41. | Write a java program for to solve producer consumer problem in which a  producer produces a value and consumer consume the value before producer generate the next value | 2 |
| 42 | **Collection and Generic Framework:**  Write a method removeEvenLength that takes an ArrayList of Strings as a parameter and that removesall the strings of even length from the list. **(Use ArrayList)** |  |
| 43 | 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 |

|  |  |  |
| --- | --- | --- |
| 44 | 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,  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. | 2 |
| 45 | **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 |
| 46 | **Java Database Connectivity (JDBC):**  **Create a database of employee with the following fields.**   * **Name** * **Code** * **Designation** * **Salary** | 2 |

|  |  |  |
| --- | --- | --- |
|  | 1. Write a java program to create GUI java application to take employee data from the TextFields and store it in database using JDBC connectivity. 2. Write a JDBC Program to retrieve all the records from the employee database. |  |
|  | Total | **48** |

1. **Suggested Books:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SL.**  **No.** | **Name of Authors/Books/Publishers** | **Edition** | **Year of Publication / Reprint** |
|  | **Textbooks** |  |  |
| 1. | Patrick Naughton and Herbert Schildt, “Java 2 The Complete Reference”, McGraw Hill Education | 9th | 2017 |
|  | E. Balaguruswamy, “Programming with Java a Primer”,  Tata McGraw Hill | 6th | **2019** |
|  | **Reference Books** |  |  |
| 1. | Core Java-Head First Publication | 3rd | 2018 |
|  | Cay S Horstmann and Gary Cornell, “Core Java Volume  –I and II”, Sun Microsystems, | Standard edition | **2019** |

|  |  |  |
| --- | --- | --- |
| 12. | **Mode of Evaluation** | Test / Quiz / Assignment / Mid Term Exam / End Term Exam |

# GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

## SEMESTER IV

Name of Department: - Computer Science and Engineering

**Microprocessors lab**

1. Subject Code: Course Title:

**PCS-403**

1. Contact Hours: **L: T**: **P**:

**2**

**1**

**0**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3. Examination Duration (Hrs): **Theo** | | | | | | | **ry 0** |  | **Practical** | **3** |  |
|  | | | | | | |  |  |  |  |  |
| 4. Relative Weight: | **CIE** |  | **25** |  |  | **MSE** | **25** |  | **SEE** | **50** |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| 5. Credits: |  |  | **2** |  |  |  |  |  |  |  |  |
| 6. Semester: |  |  | **4** |  |  |  |  |  |  |  |  |
| 7. Category of Course: |  |  | **DC** |  |  |  |  |  |  |  |  |

8. Pre-requisite: PEC151/251 Basic Electronics Engineering Lab, PCS 308 Logic design lab

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.

9. **Course Outcome\*\*:**

*\*\* Describe the specific knowledge, skills or competencies the students are expected to acquire or demonstrate*

## 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. | 1. Write an ALP in 8085 to find larger number between two numbers. 2. 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. | 1. Write a program in 8086 to add two 16-bit numbers given by the user. 2. Write a program in 8086 to subtract two 16-bit numbers given by the user. | **2** |
| 14. | 1. Write a program in 8086 to multiply two 8-bit data. 2. 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** |

1. **Suggested Books:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SL.**  **No.** | **Name of Authors/Books/Publishers** | **Edition** | **Year of Publica tion /**  **Reprint** |
|  | **Textbooks** |  |  |
| 1. | Ramesh Gaonkar, Microprocessor Architecture, Programming, and Applications with the 8085, Penram International Publication  (India) Pvt. Ltd | 6th | 2013 |
| 2. | A. K. Ray & K. M. Bhurchandi, Advanced Microprocessors and peripherals, Tata McGraw Hill | 3rd | 2012 |
|  | **Reference Books** |  |  |
| 1. | Douglas V. Hall, Microprocessors and Interfacing, Tata McGraw  Hill | 3rd | 2012 |
| 2. | Barry B. Brey, The Intel Microprocessors Architecture  Programming and interfacing, Pearson | 8th | 2012 |

|  |  |  |
| --- | --- | --- |
| 12. | **Mode of Evaluation** | Test / Quiz / Assignment / Mid Term Exam / End Term Exam |

# GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

## SEMESTER IV

Name of Department: - Computer Science and Engineering

**PCS-409**

1. Subject Code:

Course Title:

**1**

**1**

**Design and Analysis of Algorithms Lab**

1. Contact Hours: **L: T**:

**2**

**0**

**P**: **2**

1. Examination Duration (Hrs): **Theory**
2. Relative Weight: **CIE MSE**

**25**

**25**

1. Credits:

**DC**

**IV**

**2**

1. Semester:
2. Category of Course:

## Practical

**SEE**

**50**

1. Pre-requisite: PCS-302 Data Structures Lab, Any programming language

After completion of the course the students will be able to**:**

CO1: Analyze 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.:

9. **Course Outcome\*\*:**

*\*\* Describe the specific knowledge, skills or competencies the students are expected to acquire or*

*demonstrate.*

## Details of the Course:

|  |  |  |
| --- | --- | --- |
| **Sl.**  **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. | 2 |

|  |  |  |
| --- | --- | --- |
|  | 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 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   1. 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). 2. 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 |
|  | 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”. |  |
| 2 | Sample I/O Problem I: | 2 |
|  | Input: Output: |  |
|  | 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: | 1 |

|  |  |  |
| --- | --- | --- |
|  | 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  -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 total number of inversions required.  Sample I/O Problem I: Input: Output:  3 21 23 32 45 46 65 76 89 | 1 |

|  |  |  |
| --- | --- | --- |
|  | 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:  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 | 2 |

|  |  |  |
| --- | --- | --- |
|  | 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 Path Exists'. Sample I/O Problem I: | 3 |

|  |  |  |
| --- | --- | --- |
|  | 1. 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:   1. 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.   1. 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 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) 2. 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. | 3 |

|  |  |  |
| --- | --- | --- |
|  | 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 | 3 |

|  |  |  |
| --- | --- | --- |
|  | 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 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. | 2 |

|  |  |  |
| --- | --- | --- |
|  | 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  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. |  |
| 1 | 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 deadlines and also find list of selected tasks.  Input Format:  First line will give total number of tasks n. | 2 |

|  |  |  |
| --- | --- | --- |
|  | 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. 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 | **1** |

|  |  |  |
| --- | --- | --- |
|  | 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   1. 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.   1. 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 | **2** |

|  |  |  |
| --- | --- | --- |
|  | 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:  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. |  |
| 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. | **2** |
| 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:   1. 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:   1. 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** |

1. **Suggested Books:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SL.**  **No.** | **Name of Authors/Books/Publishers** | **Edition** | **Year of Publication /**  **Reprint** |
|  | **Textbooks** |  |  |
| 1. | Thomas H. Cormen, Charles E. Leiserson, Ronal L. Rivest, Clifford Stein:” Introduction to Algorithms”, 2nd Edition, PHI, | 2nd | 2006. |
|  | **Reference Books** |  |  |
| 1. | Donald E.Knuth:”The Art of Computer Programming:  Volume 1: Fundamental Algorithms”,3rd Edition | 3rd | 1997 |
| 2 | Ellis Horowitz, Sartaj Sahni, SanguthevarRajasekaran:” Fundamentals of Computer Algorithms”, 2nd Edition,  University press,. | 2nd | 2007 |

|  |  |  |
| --- | --- | --- |
| 12. | **Mode of**  **Evaluation** | Test / Quiz / Assignment / Mid Term Exam / End Term Exam |