Master of Computer Applications

Scheme and Syllabus

School of Distance Education
Andhra University, Visakhapatnam, Andhra Pradesh

M.C.A.

I/III

Sl.No.	Paper	M.C.A (previous)
1	Paper - I	Discrete Mathematical Structures
2	Paper – II	Computer Organization
3	Paper – III	Problem Solving & Programming using “C”
4	Paper – IV	Probability, Statistics & Queuing Theory
5	Paper – V	Management Accountancy
6	Paper – VI	Systems Programming
7	Paper - VII	Data Structures
8	Paper – VIII	Principals of Programming languages
9	Paper – IX	Object Oriented Programming
10	Paper - X	Information Systems & Organizational Behaviour

II/III

Sl.No.	Paper	Name of the paper
1	Paper - I	Theory of Computation
2	Paper – II	Computer Graphics
3	Paper – III	File Structures
4	Paper – IV	Design & Analysis of Algorithms
5	Paper – V	Operating Systems
6	Paper – VI	Data Communications & Networks
7	Paper - VII	Data Base Management Systems
8	Paper – VIII	Operations Research
9	Paper – IX	Artificial Intelligence
10	Paper - X	Image Processing

III/III

Sl.No.	Paper	Name of the paper
1	Paper- I	Information Systems Control & Audit
2	Paper – II	Network Security
3	Paper – III	Object Oriented Software Engineering
4	Paper – IV	Embedded Systems (Elective-II)
5	Paper - V	Data Ware Housing & Data Mining(Elective-III)

M.C.A.

Program Objective MCA

To provide students with a strong foundation in computer science and its applications, including programming languages, algorithms, data structures, database systems, software engineering, and computer networks.

To develop students' skills in critical thinking, problem-solving, and decision-making, and to enable them to apply these skills to real-world problems in computer science and related fields. To prepare students for leadership roles in the technology industry, including software development, project management, and technology consulting.To provide students with hands-on experience in software development and project management, including the use of industry-standard software development tools and techniques.To develop students' communication, collaboration, and teamwork skills, and to prepare them for working in diverse, global environments.

To provide students with a deep understanding of emerging trends and technologies in computer science, and to enable them to adapt to these changes throughout their careers. To instill in students a commitment to ethical and socially responsible behavior in the use of technology.

Overall, the objective of an MCA program is to provide students with a comprehensive education in computer science and its applications, and to prepare them for successful careers in the technology industry. The program emphasizes the development of technical skills, critical thinking abilities, and professional conduct, and provides students with opportunities to apply these skills in practical settings.

MCA FIRST YEAR

Course objectives

1.1 Discrete mathematical structures

Introduce concepts of mathematical logic for analyzing propositions and proving theorems: Mathematical logic is a branch of mathematics that studies the principles and methods of reasoning. It provides a framework for analyzing propositions and arguments, and for proving theorems using logical deduction. Some key concepts in mathematical logic include propositional logic, predicate logic, formal systems, and proof techniques such as direct proof, proof by contradiction, and mathematical induction.

Use sets for solving applied problems, and use the properties of set operations algebraically: Sets are a fundamental concept in mathematics and are used to model collections of objects. They are used in many areas of mathematics and in many applied fields such as computer science, statistics, and physics. Some important concepts in set theory include set operations (union, intersection, complement), set relations (equality, inclusion), and set properties (associativity, distributivity, De Morgan's laws).

Work with relations and investigate their properties: Relations are a way of expressing connections between objects or sets of objects. They are used in many areas of mathematics and computer science, and are particularly important in the study of functions and graphs. Some important concepts in relation theory include equivalence relations, partial orders, and functions. Properties of relations, such as transitivity, reflexivity, and symmetry, are also important in analyzing and classifying relations.

Overall, the objectives of this course are to provide students with a solid foundation in mathematical reasoning, set theory, and relation theory, which are important tools for solving problems in many areas of mathematics and beyond.

Syllabus
INTRODUCTION:

Logic-Prepositional - Equivalences-Truth-Tables-Totalogies - Predicates - and Quantifies-Sets-Operations sets-Sequences and Summations-Growth functionsrelations and their properties-n-ary relations and their applications Representation of relations-Closures of relations-Equivalence relations-Partial Orderings. Counting Techniques: Basics of Counting- Pigeonhole Principle - Combinations and Permutations - Generalized Permutations and Combinations-Recurrence relations-Solving Recurrence Relations-Divide and Conquer relations -Generating Functions-Inclusion and Exclusion-Applications of Inclusion-Exclusion

Graph Theory:

Introduction to Graphs-Terminology-Relations and Directed Graphs-Representations of Graphs-Isomorphism-Connectivity-Euler and Hamiltonian Paths-Shortest Path problems -Planar Graphs-Graph Coloring-Introduction to trees-Applications of trees-Traversals- Trees and sorting-Spanning Trees-Minimum Spanning Trees.

Boolean Algebra and Models of Computation:

Boolean Functions-Representing Boolean Functions-Logic Gates-Minimizations of Circuits-Languages and Grammars-Finite State Machines with and with no outout-Language Recognition-Turing Machines

Course objectives

1.2 computer organization

Principles and the Implementation of Computer Arithmetic: Computer arithmetic is the study of how computers perform arithmetic operations on numbers. It involves the design of circuits and algorithms that can efficiently perform addition, subtraction, multiplication, and division. Some important concepts in computer arithmetic include fixed-point and floating-point representations, overflow and underflow, and rounding errors.

Operations of CPU including RTL, ALU, Instruction Cycle, and Busses: The central processing unit (CPU) is the brain of a computer and is responsible for executing instructions. It consists of several components, including the register transfer language (RTL), the arithmetic and logic unit (ALU), the instruction cycle, and the busses that connect the CPU to other parts of the computer. Understanding the operations of the CPU is important for understanding how computers work and for optimizing performance.

Overall, the objectives of this course are to provide students with a solid understanding of the principles and implementation of computer arithmetic, as well as the operations of the CPU. This knowledge is essential for anyone interested in computer science, electrical engineering, or related fields. By the end of the course, students should be able to analyze and design computer circuits, optimize computer performance, and understand the trade-offs between different hardware architectures.

Syllabus

Digital Logic Fundamentals

Instruction Set Architectures

Introduction to Computer Organization

Register Transfer Languages

CPU Design

Micro-sequence Control Unit Design

Computer Anthmetic

Memory organization

I/O Organization

Course objectives

1.3 Problem solving and programming using 'c'

Understand problem-solving techniques: The course aims to teach students various problem-solving techniques that are commonly used in computer programming.

Use programming constructs: Students will be taught how to use conditional statements, loops, and recursion to write efficient programs. Develop algorithms: The course aims to teach students how to develop efficient algorithms to solve problems.

Write efficient code: Students will learn how to write efficient code in "C" that can be easily understood and maintained. Debug and troubleshoot: The course aims to teach students how to debug and troubleshoot their code in order to identify and fix errors. Apply programming skills: Students will be able to apply their programming skills to real-world problems.

By the end of the course, students will have a strong foundation in problem-solving techniques, programming constructs, and algorithm development using the programming language "C". They will be able to write efficient code and apply their programming skills to solve real-world problems.

Syllabus

Objective:

The objective of this subject to discuss the techniques and gotins forcing and solving various types of problems. The language used for wring programs is The emphas should be on writing algorithms and programs in Cnot merely teaching Clangage)

Introduction:

Definition of Algorithms Writing algorithms op down design-Program venfication. The efficiency of algorithms Concept of Recunion-me umple example to illustrate these concepts like finding the GCD of two numbers Swapping two variables Summation of a given numbers generation of Fibonacci sequence revening a given number-base. convention.

Introduction to C:

C character set-Delimiters The CKeywords Identifiers-Constants-Variables-Rules for Defining Variables-DosTypes-Declaring Vartables-Initializing Variables-Type Comension Ponty of Operations & their Clubbing Comma and Conditional Operator-Amimetic Operators Relational Operators -Logical Operators Bitwise Operators-Input & Output in CFormatted and Unformatted Functions Library Functions

More about C

If statement if else statement-various forms of if-nested if breakstatement-continue statement-go to statement switch statement nested switch statement fortement witlement do while statement-mays-working with string and standard functions ADVANCED CONCEPTS OFC introduction to pointers-pointer declaration-Arithmetic Operations with printers-printers and arrays-pointers and two-dimensional arrays-aray of pointers-pointers is pointer-printers and strings-void pointers-function definition and declaration-prototypes-pes of functions call by value and reference-functions returning more values-function is an argument-function with operators-function and decision statements-function and loopstatements-function with arrays and pointer-recursion pointer to function storage classes. Additionals in Copreprices directives structures and nonst wise operation files command line arguments dynamic memory allegation-graphics in C““

Problem Soving““Reversal of an Array-Removal of duplicates in an orderediary Partoning finany- Finding the ith smallest of an element of an array Finding the long montone afectence of.

Course objectives

1.4 Probability, Statistics & Queuing Theory

To provide students with a fundamental understanding of probability theory and statistical concepts. To help students develop the ability to use probability and statistics to analyze and solve problems in various fields. To teach students the basic concepts of queuing theory and its applications.

To provide students with hands-on experience in solving problems related to probability, statistics and queuing theory. To enable students to understand the importance of probability, statistics and queuing theory in various fields such as engineering, economics, finance, and management. To help students develop critical thinking and problem-solving skills using probability, statistics and queuing theory.

By the end of the course, students should be able to apply probability and statistics concepts to solve practical problems, including using appropriate software tools. They should also have a good understanding of queuing theory and its applications in real-world situations. The course should help students develop analytical and problem-solving skills and a deeper appreciation of the role that probability, statistics and queuing theory play in various fields.

Syllabus

Probability. Definitions of probability. Addition theorem, Conditional probability. Multiplication theorem, Bayes theorem of probability and Geometric probability Random variables and their properties: Discrete Random variable, Continuous Random variable, Probability Distribution joint probability distributions their properties, Transformation variables, Mathematical expectations, probability generating functions Probability-Distributions Discrete distributions. Binomial, Poisson Negative binominal distributions& their properties. (Definition, mean, variance. moment generating function, Additive properties, fitting of the distribution) Continuous distributions Uniform, Normal, exponential distributions & their properties Multivariate Analysis Correlation. correlation coefficient, Rank correlation, Regression Analysts, Multiple Regression, Attributes, coefficient of Association, c2test for goodness of fit, test for independence Estimation Sample, populations, statistic, parameter, Sampling distribution, standard error, unbiasedness, efficiency. Maximum likelihood, estimator, notion&interval.estimation Testing of Hypothesis Formulation of Null hypothesis, critic al region, level of significance, power of the test. Small Sample Tests Testing equality of means, testing equality of variances, test of correlation coefficient, test for Regression Coefficient. Large Sample tests Tests based on normal distribution

Queuing theory:

Queue description, characteristics of a queuing model, study state solutions of M/M/IA Model, M/M/1, N Model, M/M/C Model, M/M/C N Model Case Studies.

Course objectives

1.5 Management Accountancy

To provide students with an understanding of the principles and concepts of management accounting. To enable students to apply management accounting techniques to solve business problems. To teach students how to use cost accounting systems to gather, analyze, and interpret financial information for management decision making.

To help students develop the ability to prepare budgets and financial statements, and to use them to assess business performance. To provide students with an understanding of the role of management accounting in strategic planning and decision making. To enable students to appreciate the ethical considerations involved in management accounting practices.

By the end of the course, students should be able to apply management accounting principles and techniques to solve practical problems, including using appropriate software tools. They should also have a good understanding of the role that management accounting plays in strategic planning and decision making. The course should help students develop analytical and problem-solving skills and a deeper appreciation of the ethical considerations involved in management accounting practices.

Syllabus

Principles Of Accounting: Nature and scope of accounting, Double entry system of accounting, Introduction to basic books of accounts of sole proprietary concern, Closing of books of accounts and preparation of Trial Balance. Final accounts: Trading profit and loss accounts and balance sheet of sole proprietary concern with normal closing entries (with numerical problems). Ratio Analysis: Moaning, Advantages, Limitations, types of ratio and their usefulness. (Theory only) Fund flow statement: Meaning of the term fund, flow of fund, working capital cycle, Preparation and interpretation of statement. Costing: Nature, importance and basic principles. Budget and budgetary control: Nature and scope, importance, method of finalization and master budget, functional budgets. Marginal Costing: Nature, scope, importance, construction of break even chart, Limitations and uses of break-even chart, practical applications of marginal costing. (with numerical problems) Introduction to computerized accounting system: Coding Logic and codes required, master files, transaction files, Introduction to documents used for data collection, processing of different files and outputs obtained.

Course objectives

1.6 System programming

To provide students with an understanding of the fundamental concepts of system programming. To teach students how to write low-level system software that interacts directly with hardware and operating systems. To enable students to design and implement efficient algorithms for solving system-level problems.

To teach students how to use system-level programming languages and tools, such as assembly language, C, and debuggers. To provide students with an understanding of the architecture and operation of operating systems and their interfaces.To enable students to implement system-level services, such as device drivers, file systems, and network protocols.

By the end of the course, students should be able to design and implement system-level software that interacts directly with hardware and operating systems. They should have a good understanding of the architecture and operation of operating systems and their interfaces. They should also be able to use system-level programming languages and tools, and implement system-level services such as device drivers, file systems, and network protocols. The course should help students develop analytical and problem-solving skills and a deeper understanding of the principles of system programming.

Syllabus:

Introduction to grammars, languages, finite state machines, Introduction to Systems Programming, Introduction to Assembly Language Programming - Introduction to Instruction Formats, Data formats - Role of Base Register, Index Register, Introduction to Assembler, databases used in assembler design, Design of Assembler-Single Pass & Double Pass Introduction to Macros, various types of Macros, Design of Macro Processor - Single Pass & Double Pass, Introduction to Loaders, functions of a loader, types of Loaders, databases used in Loaders, Design of Loaders Absolute & DLL Introduction to compilers: a brief discussion on various-phases-of-compilers. Applications of FSM & grammars in compiler design Introduction to Software Tools, Texteditors, Interpreters, Program Generators, Debug- Monitors.

Course objectives

1.7 Data structures

Understanding of fundamental data structures: The course aims to provide students with a solid understanding of fundamental data structures such as arrays, linked lists, stacks, queues, trees, and graphs. Ability to analyze algorithms: Students will learn how to analyze the efficiency of algorithms using time and space complexity analysis.Mastery of algorithmic design: The course will teach students the principles of algorithmic design, including techniques such as divide and conquer, dynamic programming, and greedy algorithms.

Implementation of data structures and algorithms: Students will learn how to implement data structures and algorithms in code, using programming languages such as C, C++, or Java. Knowledge of data structure applications: The course will cover various applications of data structures in computer science, including searching, sorting, and graph traversal algorithms. Problem-solving skills: By working through assignments and projects, students will develop problem-solving skills in designing and implementing efficient algorithms and data structures.

Understanding of trade-offs: Students will learn the trade-offs involved in choosing between different data structures and algorithms for a given problem. Overall, the course aims to equip students with the knowledge and skills necessary to design and implement efficient algorithms and data structures for various computer science applications.

Syllabus:

Introduction to Data Structures: Abstract Data Types, Review of Strings, multi-dimesnional arrays, structures and pointer concepts in C. The Stack: Specification of ADT&primitive operators, epresenting Stacks in C, Applications of Stacks: Infix Postfix and prefix expression handling. 2. Recursion : Recursion Definition and Processes, Recursion in C, Writing Recursive Programs, Simulating Recursion, Efficiency of Recursion 3. Queues and Lists. The queues and its Sequential Representation, Linked lists, Lists in C, Circular Linked lists, Doubly linked lists. 4. Trees Binary Trees, Binary Tree Representations, Trees Their Applications, Searching: Basic Search Technologies, Tree Searching, 5. Graphs & Their Applications Graphs, Graph Traverseal & Spanning Forests, Prim's algorithm. 6. Sorting General Background, Exchange Sorts, Selection and Tree Sorting Insertion Sorts, Merge and Radix Sorts.

Course objectives

1.8 Principles of programming

Understanding of programming concepts: The course aims to provide students with a solid understanding of programming concepts such as variables, data types, control structures, functions, objects, and classes. Mastery of programming languages: Students will learn how to program using one or more programming languages such as Java, Python, C++, or JavaScript.

Ability to develop programs: The course will teach students how to develop programs, starting from problem analysis, through design, implementation, testing, and debugging. Knowledge of software development processes: The course will cover software development processes, including agile methodologies and the importance of version control systems.

Understanding of algorithmic thinking: Students will learn how to break down problems into smaller subproblems and develop algorithms to solve them. Ability to work with data structures: Students will learn how to work with various data structures such as arrays, lists, stacks, queues, trees, and graphs. Familiarity with software testing and debugging: The course will teach students how to test and debug programs to ensure their correctness.

Problem-solving skills: By working through assignments and projects, students will develop problem-solving skills in designing and implementing programs. Overall, the course aims to equip students with the knowledge and skills necessary to design and implement programs that solve real-world problems, using a systematic and disciplined approach to software development.

Syllabus:

Toward Higer-level Languages, Problems of Scale, Programming Paradigms, Language Implementation Bridging the Gap 2 Language Description: Syntactic Structure: Experssion Notations Abstract Syntax Trees, Lexical Syntax, Context-Free Grammars, Grammars for Expressions, Variants of Grammars 2

Imperative Programming:

Statements Structured Programming: The Need for Structured Programming, Syntax - Directed Control Flow Design Considerations Syntax, Handling Special Cases in Loops, Programming with invariantrs, Proof Rules for Partial Correctness, Control flow in C 4. Types: Data Representation - The Role of Types, Basic Types, Arrays Sequences of Elements Records: Named Fields, Unions and variant Records, Sets, Pointers Efficiency and Dynamic Allocation, Two String Tables, Types and Error Checking 5. Procedure Activations - Introduction to Procedures. Parameter-passing Methods, Scope Rules for Names, Nested Scopes in the Soures Text, Activation Records, Lexical Scope: Procedures as in C, Lexical Scope: Nested Procedures and Pascal

Il Object Oriented Programming:

Groupings of Data and Operations - Constructs fro Program Structuring, Information Hiding, Program Design with Modules, Modules and Defined Types, Class Declarations in C++, Dynamic Allocation I C++, Templates: Parameterized Types, Implementation of Objects in C++ 7 Object-Oriented Programming - What is an Object?, Object-Oriented Thinking. Inheritance, Object-Oriented Programming in C++ An extended C++ example, Derived Classes and information Hiding, Objects in Smalltalk, Smalltalk Objects have self

III Functional Programming:

Elements of Functional Programming - A little Language of expressions, Types Values and Operations, Function declarations, Approaches to Expression Evaluation, Lexical Scope, Type Checking 9. Functional Programming in a Typed Languages-Exp loring a list, function Declaration by Cases, Functions as First-Class Values, ML. Implicit Types, Data Types, Exception Handling In M, Little Quit in Standard ML 10 Functional Programming with Lists:- Scheme, a Dialect of Lisp, The Structure of Lists, List Manipulation, A Motivating Example: Differentiation, Simplification of Expressions, Storage Allocation for Lists.

IV Other Paradigms:

Logic Programming- Computing with Relations, Introduction to Prolog Data Structures in Pro log, Programming techniques, Control in Prolog, Cuts 12 An Introduction to Concurrent Programming - Parallelism in Hardware, Streams: Implicit Sychronization, Concurrency as interleaving, Liveness Properties. Safe Access to Shared Data, Concurrency in Ada. Synchronized Access to Sharedvaralables. TextBook Programming-Languages-Conceprts & Constructs, RaviSethi, Person Education References: 1. Programming Languages-Design & Implementation, Terrance W. Pratt, Marvin V. Zelkowitz. Pearson Education 2 concepts of Programming Languages-Robert L. Sebesta, Person Edn.

Course objectives

1.9 object oriented programming

Understanding of OOP concepts: The course aims to provide students with a solid understanding of OOP concepts such as encapsulation, inheritance, polymorphism, abstraction, and classes. Mastery of programming languages: Students will learn how to program using OOP languages such as Java, Python, C++, or Ruby.

Ability to develop object-oriented programs: The course will teach students how to develop programs using object-oriented programming principles and design patterns. Familiarity with design patterns: The course will cover various design patterns such as singleton, factory, and observer patterns, which are commonly used in OOP.

Knowledge of software design principles: The course will cover software design principles such as SOLID (Single Responsibility, Open-Closed, Liskov Substitution, Interface Segregation, and Dependency Inversion) that promote good software design practices. Ability to work with OOP features: Students will learn how to work with OOP features such as classes, objects, inheritance, polymorphism, and interfaces. Familiarity with testing and debugging OOP programs: The course will teach students how to test and debug OOP programs to ensure their correctness.

Problem-solving skills: By working through assignments and projects, students will develop problem-solving skills in designing and implementing OOP programs. Overall, the course aims to equip students with the knowledge and skills necessary to design and implement OOP programs that are modular, maintainable, and scalable, using software design principles and design patterns.

Syllabus:

Fundamentals of object oriented programming Introduction to Object Oriented Paradigm, Procedural Paradigm, An overview of classes, objects and Methods, Inheritance and polymorphism. Basic of C++ Structure of C++ program, data types and declaration, Expressions and operator precedence, Program flow control, functions, scope of variables, Inline functions and default arguments, dynamic allocation new and delete operators: Classes as objects, user defined data types, constructors & destructors, controlling and accessibility, class members, member functions, Friend functions, this pointer, static and const member functions, inheritance Derived classes, syntax of derived classes, Types of Inheritance. Virtual Functions, and Virtual Base Classes Adhoc Polymorphism: Overloading and Function selection, Friend Functions, overloading operators such as assignment subscripting, VO, pointer to class member, new and delete Templates Generic Classes, Class Templates, Function Templates Parameterizing Vectors, STL, Containers, Iterators, Function Adapters, String Library Exceptions: Using asserth, signal h, throwing exceptions, Try Blocks, handlers, Exception specification standard exceptions and uses. I/O streams: Output and Input class streams, Ostream, Istream, File handling, using strings as streams UML Basics, Use Case, Class, Object, Sequence, Activity, State Chart, Collaboration, Component and Deployment diagrams in Object oriented project design.

Course objectives

1.10 Infromation System & Org. Behaviour

Understanding of the relationship between information systems and organizations: The course aims to provide students with an understanding of how information systems can be used to support and improve organizational performance.

Knowledge of organizational behaviour: Students will learn about organizational behaviour theories and concepts such as motivation, leadership, communication, and decision making. Familiarity with information systems: The course will cover various types of information systems such as transaction processing systems, decision support systems, and enterprise resource planning systems.

Understanding of the role of information systems in organizations: The course will explore how information systems can be used to support various organizational functions such as finance, human resources, marketing, and operations.

Ability to analyze organizational issues: Students will learn how to analyze organizational issues and problems, and how information systems can be used to address these issues. Knowledge of ethical and social issues related to information systems: The course will cover ethical and social issues related to the use of information systems, such as privacy, security, and intellectual property. Communication skills: Students will develop communication skills by working on team projects and presenting their findings to the class.

Problem-solving skills: By working on case studies and projects, students will develop problem-solving skills in analyzing organizational issues and designing information systems to address them. Overall, the course aims to equip students with the knowledge and skills necessary to understand the relationship between information systems and organizations, and how information systems can be used to support organizational performance and decision making. The course will also explore ethical and social issues related to the use of information systems in organizations.

Syllabus:

Organizational Structure and Design - Managerial Communication and its barriers Controlling - Delegation of Authority and Inter Departmental Co-ordination Organizational Climate and Culture - Management of Organizational Conflicts- Theories of Motivation Group Dynamics-Characteristics of a Leader-Leadership Styles-Analysis of Interpersonal Relations MIS Perspective - Information needs and its objectives - Management Information and Control Systems Information for Decision Making- Conceptual Foundations of Information Systems - Information Resource Management.

Course objectives

1.11 computer organization lab

Understanding of computer organization concepts: The course aims to provide students with an understanding of computer organization concepts such as memory hierarchy, instruction set architecture, and processor design. Familiarity with computer hardware: Students will learn about computer hardware components such as CPUs, memory, and input/output devices.

Ability to assemble and configure computer systems: The course will teach students how to assemble and configure computer systems, including installing operating systems and device drivers. Knowledge of programming languages: The course will cover programming languages such as assembly language, C, and C++, which are used to develop low-level programs that interact with computer hardware.

Familiarity with computer simulation tools: The course will introduce students to computer simulation tools such as SPIM and MARS, which are used to simulate computer architecture and programming. Ability to design and implement low-level programs: Students will learn how to design and implement low-level programs such as device drivers, interrupt handlers, and system utilities.

Familiarity with debugging and profiling tools: The course will cover debugging and profiling tools such as GDB and Valgrind, which are used to debug and optimize low-level programs. Problem-solving skills: By working on lab assignments and projects, students will develop problem-solving skills in designing and implementing low-level programs that interact with computer hardware.

Overall, the course aims to equip students with the knowledge and skills necessary to understand computer organization concepts and design and implement low-level programs that interact with computer hardware. The course will also introduce students to computer simulation and debugging tools, which are essential for low-level programming.

Syllabus:

Cycle –I Computer Organization Laboratory

01	TTL Characteristics
02	TTLIC Gates
03	Implementation of Gates
04.	Realisation of Gates
5.1	Flip-Flops
5.2	R-S Filp Flops
5.3	D Filp Flops
5.4	T Flip Flops
06.	JK Flip Flops
07.	Counters
08.	Shift Registers & Design of Shift Register
09.	Multiplexers
10.	a) Decoders b) 2-4 Line Decoder c) 3-8 Line Decoder
11.	Encoder
12.	Parity Checker
13.	Adders & Sub tractors
14.	Gray to Binary & Binary to Gray
15.	Seven Segment Display
16.	Design of MOD-10 Counter

Cycle-ll: Micro Processor Laboratory

17.	Addition of 8 bit numbers
18.	Subtraction of 8 bit numbers
19.	Multiplication of 8 bit numbers
20.	Division of 8 bit numbers
21.	Factorial of given number
22.	Block transfer to Data
23.	Searching of an element in array
24.	Counting number 1’s in a given number
25.	Maximum Element in an Array
26.	Minimum Element in an Array
27.	Addition of two 18-bit numbers

28. Subtraction of two 16-bit numbers
29. Sorting away in ascending order
30. Sorting array in descending order

Course objectives

1.12 'C' programming lab

Understanding of programming concepts: The course aims to provide students with an understanding of programming concepts such as data types, control structures, functions, and arrays. Mastery of the C programming language: Students will learn how to program in the C language, including syntax, semantics, and common library functions.

Ability to write efficient code: The course will teach students how to write efficient code using techniques such as loop unrolling, function inlining, and cache optimization. Familiarity with debugging tools: The course will cover debugging tools such as GDB, which are used to debug C programs.

Knowledge of data structures: The course will introduce students to data structures such as linked lists, stacks, and queues, and how to implement them in C. Ability to design and implement algorithms: Students will learn how to design and implement algorithms such as sorting and searching algorithms, and how to evaluate their performance.

Familiarity with software development tools: The course will cover software development tools such as make and version control systems, which are used to manage large software projects. Problem-solving skills: By working on lab assignments and projects, students will develop problem-solving skills in designing and implementing C programs that solve real-world problems.

Overall, the course aims to equip students with the knowledge and skills necessary to program efficiently in the C language, design and implement algorithms, and work with software development tools. The course will also introduce students to data structures and debugging tools, which are essential for programming in C. By working on lab assignments and projects, students will develop problem-solving skills in designing and implementing C programs.

Syllabus:

01. Introduction to Programming
02. Swapping Two Variables..
03. Sum of Digits of a given Number.
04. Reversing a Given Number
05. Finding Maximum and Minimum of a given Array
08. Solving a Quadratic Equation.

07. Operations Based on Menu.
08. Sum to N terms of a Sine Series
09. Matrix Multiplication
10. Transpose of a Matrix.
11. Polynomial Addition
12. Polynomial Multiplication
13. Sorting
14. Searching.
15. Finding Length of a String
16. Reverse of a String.
17. Concatenation of Strings
18. Palindrome Checking
19. Lower to Upper Case and Vice Versa.
20. Sum of the Numbers of an Array using Pointers.
21. Swapping Contents Using Pointers
22. Finding the first and Second Ranks Using Structures.
23. Complex Numbers as Structures
24. Unions
25. Copying and Concatenation of Files
26. Bitwise Operations.
27. Command Line Parameters.
28. Preprocessor Directives
29. Macros
30. Recursion, Merge Sort and Quick Sort.

Course objectives

Object oriented programming lab

To understand the principles of object-oriented programming and apply them to real-world problems. To learn how to design, implement and test object-oriented software solutions. To gain proficiency in a specific object-oriented programming language such as Java, Python, or C++.

To learn how to use object-oriented programming tools and frameworks such as Eclipse or Visual Studio. To develop problem-solving skills through the use of object-oriented programming concepts such as encapsulation, inheritance, and polymorphism. To gain experience in writing and debugging object-oriented code.

To learn how to work collaboratively in a team to develop and deliver software projects. To gain an appreciation for the importance of software engineering principles such as code reusability, maintainability, and scalability. Overall, the objective of an OOP lab is to provide students with hands-on experience in designing, implementing, and testing object-oriented software solutions, which will prepare them for careers in software development or further studies in computer science.

Syllabus:

Object Oriented Programming Lab

List of experiments:

1. Illustrato passing by Reference (Programme 4.6)
2. Ilustrze use of static inside a class. (Programme 4.7)
3. Demonstrate - usage of Friend Function(Prog.4.9)
4. Demonstrate Friend Class (Programme 4.10)
5. Complex No.s adding and multiplying (Prog. 4.13)
6. Copy constructor demo (Programme. 5.8)
7. User defined copy constructor demo (Prog 5.9)
8. Operator +, overloading (Prog.6.11, 6.12)
9. Adding - Rational-Numbers (Prog. 6.13)
10. Overloading-Auto- increment-operator.(Prog.6.14)
11. InteractiveConstructor(Prog.7.4)
12. Real Time Digital Clock (Programme 8.9, 9.1)
13. Virtual base class Demo (Programme 9.2
14. Is-a', 'has-a' relationships (Programme 9.4, 9.5)
15. Polymorphism using Pointer to Object (Programme 12.2)
16. Virtual base class Demo (Programme 12.9)
17. Binary File Demo (Programme 13.7)
18. Creating large file (Programme 13.12
19. File split, File jolin (Prog 13.13, 13.14)
20. Template sorting (Prog.14.4)
21. Demo of Class Template (Prog. 14.5)
22. Matrix Multiplication (Prog 15.3)

23. Linked list implementation (Prog. 15.16, 15.17, 15.18)

24. Stack simulation (Prog. 15.19

25. Demo of using keyword CONST (Prog. 16.8, 16.10)

26. Drawing lines (Prog 17.4)

27. Storing image on Disk (Prog. 17.9)

28. Animation (Prog. 17.10)

29. Using Mouse (Prog. 17.11)

30. Visual Basic form creation (Prog. 17.12) Ref Objec Oriented Programming with C++, M.P. Bhave and S.A. Patekar, Pearson Education.

Course objectives

1.14data structure lab

To understand the concepts of data structures such as arrays, linked lists, stacks, queues, trees, and graphs. To learn how to implement various data structures using a programming language such as C, C++, or Java. To gain proficiency in using algorithms to manipulate and search data structures.

To develop problem-solving skills through the use of data structures to solve real-world problems. To learn how to analyze the time and space complexity of algorithms and data structures. To gain experience in writing efficient algorithms for manipulating data structures.
To learn how to use data structures to implement common programming tasks such as sorting, searching, and graph traversal. To gain an appreciation for the importance of choosing the right data structure for a given problem.

Overall, the objective of a data structure lab is to provide students with hands-on experience in designing, implementing, and analyzing data structures and algorithms, which will prepare them for careers in software development or further studies in computer science.

Syllabus:

Data Structures Lab

List of Experiments: 1. ADT Stack implementation and use it for evaluation of post-fix expression 2. Conversion of Prefix expression into Postfix form using recursion 3. Implementation of circular queue (using array) with menu options like insert, delete, display and exit. 4 Implementation of a priority queue (using pointers) & use it to organize student records prioritised by marks 5. Implementation of ADT doubly linked circular list to hold strings and use it for organizing a sequence of cities constituting a tour program. 6.Implementation of a binary search tree with menu options Construct a tree, insert a node, delete a node, traverse and display preorder, inorder and post order sequence of its nodes. 7. Implementation of di-graphs using adjacency matrix and find the transitive closure using Warshall’s algorithm. 8. Implementation of a weighted graph and find minimal cost spanning tree using PRIM’s Algorithm 9. Generale 100 random integers in a given range and sort them using quick sort. Apply both binary search and Interpolation search to locate a given integerand compare the search algorithms based on the number of comparisons/probes required for a successful as well as unsuccessful search 10. Heap Sort. 11 Merge Sort. 12. Implementation of a small Real World Application illustrating DS usage.

MCA SECOND YEAR

Course objectives

2.1 Theory of Computation

To understand the fundamental concepts of automata theory and formal languages. To learn how to use mathematical models such as regular expressions, context-free grammars, and Turing machines to describe and analyze formal languages. To gain proficiency in proving theorems related to automata theory and formal languages.

To understand the Church-Turing thesis and its implications for computability theory. To learn about complexity theory and the classes of problems that are solvable by computers. To gain an appreciation for the limitations of computation and the existence of unsolvable problems. To understand the relationships between different models of computation such as Turing machines, pushdown automata, and finite automata.

To gain experience in solving problems related to automata theory and formal languages. Overall, the objective of a Theory of Computation course is to provide students with a deep understanding of the mathematical foundations of computer science, which will prepare them for advanced studies in computer science, artificial intelligence, and other related fields.

Syllabus:

Introduction to Finite Automata Alphabets and languages-Finite Representation of Languages. Deterministic Finite Automata-Non-deterministic Finite Autometa-Equivalence of Deterministic and Non-Finite Automata Properties of the Languages Accepted by Finite Automata- Finite Automata and Regular Expressions-Proofs those Languages Are and Are Not Regular 2 Context free languages Context-Free Grammar-Regular Languages and Context Free Grammar-Pushdown Automata- Pushdown Automata and Context-Free dammar-Properties 01 Context-Free Languages-Closure Properties-Periodicity Properties-Determinism & Parsing-Deterministic Pushdown Automata and Context-Free Languages-Top-down Parsing-Bottom-Up parsing 3 Turing machines The Definition of Turing Machine-Computing with Turing Machines-Combining Turing Machines-some Examples of More Powerful Turing Machines 4 Church' Thesis Church's Thesis-The Primitive Recursive functions-Godelization - The m-Recursive Functions-Turing-Computability of the m-Recursive functions-Universal Turing Machines.

5. Uncomputability The Halting Problem-Turing-Enumerability, Turing-Acceptability, Turing-Decidability- Unsolved problems about Turing machines and m- Recursive Functions-Post's correspondence problem 6 Computational complexity Time-bounded Turing Machines-Rate of Growth of functions-Time-Bounded Simulations-The Classes P and NP- NP-Completeness-Some NP-complete Problems-Integer Programming-The Traveling Salesman Problem 7 The Prepositional Calculus Introduction-Syntax of the Prepositional Calculus-Truth-Asignments Validity and Satisfiability-Equivalence and Normal Forms-resolution in Prepositional Calculus 8 The predicate calculus Syntax of the Predicate Calculate Calculus-structures and Natisfiability-Equivalence-Unsolvability & NP Completeness-Resolution in the

Predicate Calculus Test Hooks Blements of The Theory Of Computation Harry R Lewis, Cristos h. Papadimitriou, Peran Bducation Prentice-Hall of India Private Limited Reference: Introduction to Automata Thery, Languages & Computation, Hopcroft. J.E&JD. Ullman. Addison-Wesley, Reading, Mass. 1979

Course objectives

2.2 **Computer Graphics**

To understand the fundamental concepts of computer graphics and its applications. To learn how to represent 2D and 3D objects in a computer using mathematical models such as points, lines, and surfaces. To gain proficiency in using programming languages such as OpenGL or DirectX to create interactive computer graphics applications.

To learn how to apply various algorithms and techniques for rendering, shading, and lighting in computer graphics. To understand the principles of computer animation and how to create animations using keyframe and motion capture techniques. To gain experience in designing and implementing 2D and 3D graphical user interfaces.

To learn how to use computer graphics tools such as Adobe Photoshop, Illustrator, and Maya. To gain an appreciation for the ethical and social implications of computer graphics in fields such as art, entertainment, and medicine.

Overall, the objective of a Computer Graphics course is to provide students with a comprehensive understanding of computer graphics, which will prepare them for careers in computer graphics programming, game development, animation, and other related fields.

Syllabus:

Introduction Usage of Graphics and their applications, Presentation Graphics Computer Aided Design Computer Art Entertainment Education and Training Visualization-Image Processing Graphical User Interfaces Over view of Graphics systems: Video Display Devices-Raster Scan systems-randont scan systems-Graphics monitors and workstations Input devices-hard copy devices-Graphics software Output primitives: Points and Lines-Line Drawing Algorithms-Loading the Frame buffer-Line function Circle- Generating Algorithms- Empse Generating Algorithms-Other Curves Parallel Curve Algorithme Curve Functions-Pirel Addressing-Filled Area Primitives-Filed Area Functions- Cell Array Charecter Generation Attributes of Output Primitives Line& Curve Altributes-Color and Gray scale levels-Area Fill Attributes-Character Attributes-Bundled Attributes- Inquiry Functions-Antialiasing Two-Dimensional Geometric-Transformations Basic-Transformations- Matrix-Representations-Homogeneous Coordinates Composite Transformations-Other Transformations-Transformations between Coordinate Systems-Affine Transformations- Transformation Functions- Raster methods for Transformations Two Dimensional Viewing: The viewing Pipeline Viewing Coordinate Referenceframe-Window-to-Viewport Coordinate-Transformation- Two Dimensional Viewing Functione-Clipping Operations-Point Clipping-Line Clipping-Polygon

Clipping-Curve Clipping-Text and Exterior Clipping Structure And Hierarchical Modeling Concepts of Structures and Basic models- Editing-Hierarchical Modeling with Structures-GUI and Interactive Input Methode- Windows and Icons- Virtual Reality Environments Three Dimensional Concepts and Object representations: 30 display methods-3D Graphics-Polygon Surfaces-Curved Lines and Surfaces-Quadratic Surfaces-Super Quadrics-Blobby Objects-Spline Representations-Cubic Spline methods-Bézier Curves and Surfaces- B Spline Curves and Surfaces Three Dimensional Geometric& Modeling Transformations: Translation- Rotation-scaling-Other Transformations- Composite Transformations-3D Transformation Functions-Modeling and Coordinate Transformations Three Dimensional Viewing: Viewing Pipeline-Viewing Coordinates-Projections-View Volumes- General Projection Transformations-Clipping-Hardware Implementations-Three Dimensional Viewing

Course objectives

2.3 File Structures

To understand the concepts of file organization and file processing. To learn how to design and implement different types of file structures such as sequential, indexed, and direct access files. To gain proficiency in using file management systems and file access methods.

To learn how to analyze the performance of file structures and access methods. To understand the principles of database systems and their relationship to file structures. To gain experience in using file structures to solve real-world problems such as data storage and retrieval. To learn how to use different programming languages such as C, C++, or Java to implement file structures and access methods. To gain an appreciation for the importance of data integrity, security, and backup in file processing.

Overall, the objective of a File Structures course is to provide students with a solid understanding of file processing and file management systems, which will prepare them for careers in software development, database administration, and other related fields.

Syllabus:

File Processing Operations Physical and logical files, opening, reading & writing and closing files in C. secking and special characters in files, physical devices and logical files, file-related header files in CSecondary Storage Disks-organization, tracks, sectors, blocks, capacity, non-data overhead, cost of a disk access. Magnetic Tape-types, performance, organization stimation of tape length and data transmission times, disk vs tape, CD-ROM-CD-ROM as afile structure, physical organization, strengths and weakness of CD-ROMS, storage hierarchy Byte Journey and buffer Management File man ager, I/O buffer, I/O processing, buffer strategies and bottlenecks File Structure Concepts A stream file, field structures, reading a stream of fields, record structures and that uses a length indicator, Mixing numbers and characters-use of a hex dump reading the variable length records from the files Managing records in C files Retrieving records by keys, sequential search, direct access, choosing a record structure and record length, header records, file access & file organization Organizing files for performance Data compression, reclaiming space-record deletion and storage compaction, deleting fixed length records for reclaiming space dynamically, deleting variable-length records, space fragmentation, replacement strategies. Indexing Index, Asimple index with an entry sequenced file, basic operations on an indexed, entry sequenced file, indexes that are too large to hold in memory, indexing to provide access by multiple keys, retrieval using combination of secondary keys, improving the secondary index structure-inverted lists Indexed sequential file access and prefix B+Trees Indexed sequential access, maintaining a sequence set, adding a simple index to the sequence set, the content of the index: separators instead of keys, the simple prefix B-tree, simple prefix B+ tree maintenance, index set block size, internal set block size, internal structure of index set blocks: a variable order B-tree, loading a simple prefix B+ Hashing Collisions in hashing, a simple hashing algorithms, hashing functions & record distributions, memory requirements, collision resolution by progressive overflow,buckets, deletions Extendable hashing Working of extendable bashing.Implementation, deletion, exersable hasking performance Dodgning file structure for CD-ROM Tree structure on CD-ROM, hasbing flies on cu ROM CD-ROM file structure Implementation in C++ Test Book Fe Sowsures An Object Oriented Approach with C++, Michael J.Folk, Bill- Zoeck & Greg Kioward, Person Edn.

Course objectives

2.4 Design & Analysis of Algorithms

To understand the fundamental concepts of algorithm design and analysis. To learn various algorithm design techniques such as divide and conquer, dynamic programming, and greedy algorithms. To gain proficiency in analyzing the time and space complexity of algorithms.

To learn how to evaluate the performance of different algorithms for solving the same problem. To gain an understanding of the theory of NP-completeness and the relationship between different classes of problems.

To learn how to apply algorithmic techniques to solve real-world problems such as data compression, cryptography, and network routing. To gain experience in implementing and testing algorithms using a programming language such as C, C++, or Java. To gain an appreciation for the ethical and social implications of algorithm design and analysis.

Overall, the objective of a Design and Analysis of Algorithms course is to provide students with a deep understanding of the theoretical foundations of computer science and practical skills for designing efficient algorithms, which will prepare them for careers in software development, data science, and other related fields.

Syllabus:

Introduction Notion of Algorithan - Algorithmic Problem solving (1.1, 1.2) 2 Analysis of Algorithm Efficiency Analysis framework Asymptotic notations Analysis of Non-recursive and recursive algorithms (2.1, 2.4) 3.Divide and Conquer. Merge sort. Quick Sort, Binary search, Large integer Multiplication and Strassens Matrix multipli-cation- closest pair and convex Hull problems (4.1 to 4.3, 45, to 4.6) 4. Decrease & conquer DFS & BFS, opological sorting. Decrease by a Constant factor Algorithms, variable size Decrease Algorithms (5.2 to 5.6) 5. Transform and conquer Homer's Rule and Binary Exponentiation Problems Reduction (6.5, 6.6) 6. Space & Time Trade offs- Input Enhancement in String Matching (7.2) 7. Dynamic Programming Warshall's Floyd's Algorithm - Optional Binary Search Trees knapsack Problem (8.2 to 8.4) 8. Greedy Technique Print's and kruskal's Algorthims, Dijkstra's Algorithm, Huffman Trees (9.1 to 9.4) 9. Limitations of Algorithm Power Lower Bound Arguments Decision Trees-P, NP and NP Complete problems (10.1 to 10.3) 10. Coping with limitations of Algorithmic Power Backtracking. Branch & Bound, Approximation Algorithms for NP hard problems (11.1 to 113)

Course objectives

2.5 Operating systems

To understand the fundamental concepts of operating systems and their role in computer ystems.
To learn how operating systems manage system resources such as memory, input/output devices, and processors.
To gain proficiency in programming with system-level APIs for operating systems such as POSIX, Win32, or macOS APIs.
To learn how to implement basic operating system components such as process scheduling, memory management, and file systems.
To understand the principles of concurrency and synchronization in operating systems.
To gain an understanding of virtualization techniques and their role in modern operating systems.
To learn about the design and implementation of distributed systems and their relationship to operating systems.
To gain an appreciation for the ethical and social implications of operating systems in fields such as cybersecurity, privacy, and data governance.
Overall, the objective of an Operating Systems course is to provide students with a comprehensive understanding of operating systems and their role in modern computing, which will prepare them for careers in operating system development, cybersecurity, cloud computing, and other related fields.

Syllabus:

Over View Introduction: Computer System Structures. Operating Systems structures Process Management Processes, Threads, CPU scheduling. Process synchronization, Deadlocks Storage Management Memory Management. Virtual memory, file system, I/O systems, Mass storage structure Protection and Security Protection and Security.

Course objectives

2.6 Data Communication & Networks

To understand the fundamental concepts of data communication and networking. To learn about the different types of network topologies and protocols used in data communication. To gain proficiency in programming with network APIs such as sockets in C, C++, or Java. To learn about the different layers of the OSI model and their corresponding protocols. To understand the principles of TCP/IP and how it is used in modern networking. To gain an understanding of network security concepts such as encryption, authentication, and access control.

To learn about the design and implementation of different types of networks such as LANs, WANs, and wireless networks. To gain an appreciation for the ethical and social implications of networking in fields such as cybersecurity, privacy, and data governance. Overall, the objective of a Data Communication & Networks course is to provide students with a comprehensive understanding of networking technologies and their role in modern computing, which will prepare them for careers in network administration, cybersecurity, cloud computing, and other related fields.

Syllabus:

1. Introduction: Data Communications, Networks, The Internet, Protocol & Standards.

2. Network Models: Layered tasks, Internet model, OSI model

3. Physical Layer: 3.1 Signals:

Analo and digital signals, data rate limits, Transmission Impairment, Signal measurements like throughput, propagation speed and time, wave length 3.2 Digital Transmission: Line coding, block codin, sampling, transmission mode 3.3. Analog Transmission Modulation digital data, telephone modem, Modulation analog signals 3.4. Multiplexing FDM, WDM TDM 3.5 Transmission Media: Guided media, unguided media 3.6. Circuit Switching & Telephone Network: Circuit switching.telephone network.

4. Data Link Layer: 4.1 Error detection and Correcttion: Type of errors, detection and correction of errors 4.2 Data Link Control & Protocol Flow & Error control, Stop-And- Wait ARQ Go-Back-N ARQ, Select Repeat ARQ, HDLC 4.3. Point-to-Point Access Point-to-Point Protocol, PPP Stack 4.4. Local Area Network Traditional Ethernet, Fast and gigabit Ethernets 4.5 Connecting LANS Backbone Networks and Virtual LANs: Connecting devices, Backbone Networks, Virtual LANS5. Network Layer: 5.1 Internetworks, Addressing, Routing, 5.2. Network Layer Protocols ARP, IP, ICMP, IPV6 5.3. Unicast routing, Unicast routing protocols, Multi routing, Multicast routing protocals

6. Transport Layer: 6.1. Process-to-Process delivery user datagram, Transmission control protocol

7. Application Layer: 7.1 Client-Server Model: Client-Server Model, Socket interface

7.2. A brief introduction to DNS, SMTP, FTP

Course objectives

2.7 Data base management system

To understand the fundamental concepts of database management systems and their role in modern computing. To learn about different types of databases such as relational, NoSQL, and graph databases. To gain proficiency in using Structured Query Language (SQL) for database manipulation and retrieval.

To learn about database design principles and normalization techniques. To understand the principles of transaction management and database concurrency control. To gain an understanding of database security concepts such as authentication, authoriation, and encryption.

To learn about the design and implementation of different types of database systems such as distributed databases and data warehouses. To gain an appreciation for the ethical and social implications of database management systems in fields such as privacy, data governance, and social justice.

Overall, the objective of a Database Management System course is to provide students with a comprehensive understanding of database technologies and their role in modern computing, which will prepare them for careers in database administration, data science, software development, and other related fields.

Syllabus:

1. Database Systeins Concepts And Architecture:Introduction, data models, schemas and instance; three-schema architecture and data independence; database language and interface, the database systern environment, centralized and client/server architecture of DBMSs; classification of DBMSs.

2. Data Modeling Using The E-R Model: High-level conceptual data models for database design, Entity types, entity sets, atinbutes and keys, relationship types, relationship sets, roles and structural constraint, weak entity types, ER diagrams, naming conventions and design issues; Notation for UML class diagrams

3. Enhanced ER And UML Modeling: Subclasses, super classes and inheritance; specialization and generalization; constraints and characteristics of specialization and generalization, modeling of union types using categories; representing specialization/ generalization and inheritance in UML class diagrams; relationship types of degree higher than two, data abstraction, knowledge representation and ontology concepts

4. The Relational Data Model And Relational Database Constraints: Relational model concepts, relational model constraints and relational database schemas; updating operations and dealing with constraints violations

5. The Relational Algebra And Relational Calculus Unary relational operations: SELECT and PROJECT, relational algebra operations from set theory, binary relational operations: JOIN and DIVISION; additional relational operations; the tuple relational calculus; the domain relational calculus.

6. Relational Database Design By ER And EER-To-Relational Mapping: Relational database design using ER-to-Relational mapping, mapping EER model constructs to relations.

7.Functional Dependencies And Normalization For Relational Databases: Informal design guidelines for relational schemas, functional dependencies, normal forms based on primary leys, general definitions of 2nd and 3rd normal forms, Boyce-Codd normal forms

8. Transaction Processing Concepts: Introduction to transaction processing, transaction and system concepts, desirable properties of transaction, characteristics schedule based on recoverability; characteristics schedule based on serializability.

9.Concurrency Control Techniques: Two phase locking techniques for concurrency control; concurrency control based on time stamp ordering, multi-version concurrency control techniques; validation(optimistic) concurrency control techniques granularity of data items&multi granularity locking.

10. Database Recovery Techniques. Recovery concepts, recovery techniques based on deferred updates;recovery techniques based on immediate update, shadow paging the ARIES recovery algorithm. Text Book: Fundamentals of Database Systems RamezElmasri & Shamkant B.Navathe, Pearson 4 edn. Ref: Database Concepts, Abraham. Silberschatz, Henry F.Korth, S.Sudarshan, TMGH.

Course objectives

2.8 Operation Research

To understand the fundamental concepts of operations research and its applications in decision-making. To learn about different optimization techniques such as linear programming, integer programming, and nonlinear programming. To gain proficiency in using optimization software such as Excel Solver or MATLAB Optimization Toolbox.

To learn about probabilistic modeling and statistical inference techniques used in operations research. To understand the principles of queuing theory and its applications in service systems. To gain an understanding of simulation modeling and its applications in analyzing complex systems. To learn about decision analysis techniques such as decision trees and game theory.

To gain an appreciation for the ethical and social implications of operations research in fields such as sustainability, social justice, and public policy. Overall, the objective of an Operations Research course is to provide students with a comprehensive understanding of optimization and decision-making techniques and their role in modern computing, which will prepare them for careers in operations research, data science, supply chain management, and other related fields.

Syllabus:

Overview of operations Research OR models-OR Techniques Linear Programming: Introduction- Graphical solution, Cmphical sensitivity analysis-The standard form of linear programm...ig problems- Basic feasible solutions-unrestricted variables-simplex algorithm-artificial variables-Big M and two phase method-Degeneracy-alternative.ptima-unbounded solutions-infeasiblesolutions. Dual problems Relation between primal&dual problems-Dual simplex méthod Transportation model: starting solutions North West corner Rule-lowest cost method-Vogels approximation method- Transportation algorithms-ssignment problem-Hungarian Method Network.Models: Definitions CPM&PERT,their Algorithms Integer Programming Branch&Bound Algorithms cutting plan algorithm. Dynamic Programming Recursive nature of dynamic programming-Forward and Backward Recursion Deterministic Inventory Models: Static EOQ Models-Dynamic EOQ models.Game theory: Two person Zero Sum Games-Mixed strategy games& their Algorithms. Text Books: 1.Operations Research- An Introduction, Handy ATaha-Pearson Edn. [Chapter! to 11,14]2.Operations Research Panneer Selvan PHI

Course objectives

2.9 Artificial Intelligence

To understand the fundamental concepts of AI and its applications in modern computing. To learn about different AI techniques such as machine learning, natural language processing, and computer vision. To gain proficiency in using AI libraries and frameworks such as TensorFlow, Keras, and PyTorch.

To learn about different types of machine learning such as supervised, unsupervised, and reinforcement learning. To understand the principles of deep learning and its applications in computer vision and natural language processing. To gain an understanding of symbolic reasoning and its applications in expert systems.

To learn about the ethical and social implications of AI in fields such as privacy, data governance, and social justice. To gain hands-on experience in developing AI applications and analyzing real-world datasets.

Overall, the objective of an Artificial Intelligence course is to provide students with a comprehensive understanding of AI technologies and their role in modern computing, which will prepare them for careers in AI research, data science, software development, and other related fields.

Syllabus:

1. Problems and Search: What is Artificial Intelligence?, The Al Problems. The Underlying Assumption, What is an AI Technique, The Level of the Model, Criteria for Success, Some General References, One Final World..

2. Problems, Problem Spaces and Search: Defining the Problem as a State Space Search, Production systems, Problem Characteristics, Production System Characteristics, Issues in the Design of Search Programs, Additional Problems.

3. Heuristic Search Techniques: Generate-and-Test, Hill Climbing, Best-First Search, Problem Reduction, Constraint Satisfaction, Means-Ends Analysis.

4. Knowledge Representation: Knowledge Representation Issues, Representations and Mappings, Approaches to knowledge Representation, Issues in Knowledge Representation, The Frame Problem.

5. Using Predicate Logle: Representing Instance and Isa Relationships, Computable Functions and Predicates, Resolution, Natural Deduction.

6. Representing Knowledge Using Rules: Procedural Versus Declarative knowledge, Logic Programming, Forward versus Back Ward Reasoning, Matching, Control knowledge.

7. Symbolic Reasoning under Uncertainty: Introduction to Nonmonotonic Reasoning, Logics for Nonmonotonic Reasoning. Implementation Issues, Augmenting a problem solver, Implementation: Depth-First Search, Inplementation: Breadth First Search.

8. Statistical Reasoning: Probability and Baye’s Theorem, Ginty Factors and Rule-Based Systems, Bayesian Networks, Dempster-Shafer Theory zzy Logic.

9. Weak Slot-and-Filler Structure: Semantic Nets, Frames.

10. Strong Slot-and Filler Structure: Ceceptual Dependency $ripts, CYC.

11. Knowledge Representation Surnur Syntactic-Semantic Logic and Slot-and-Filler Structures, Other Representational Role of Knowledge.

Course objectives

2.10 Image Processing

To understand the fundamental concepts of image processing and its applications in modern computing. To learn about different techniques for image enhancement, restoration, and segmentation. To gain proficiency in using image processing libraries and frameworks such as OpenCV and MATLAB Image Processing Toolbox.To learn about different types of image analysis such as pattern recognition and computer vision.

To understand the principles of deep learning and its applications in image classification and object detection. To gain an understanding of the ethical and social implications of image processing in fields such as privacy, data governance, and social justice.To gain hands-on experience in developing image processing applications and analyzing real-world image datasets.

Overall, the objective of an Image Processing course is to provide students with a comprehensive understanding of image processing technologies and their role in modern computing, which will prepare them for careers in image processing research, computer vision, robotics, and other related fields.

Syllabus:

1. Fundamentals of Image Processing: Image Acquisition, Image Model, Sampling Quantization, Relationship between pizdis, distance meas, connectivity, Image Geometry, Photographic film. Histogram: Defintion, decision of commst basing on histogram, operatio na basing on histogramas like image stretching, image sliding, Image classification Definition and Algorithm of Histogram equalization.

2. Image Transforms: A detail discussion on Fourier Transform, DFT, FFT, properties A brief discussion on WALSH Transform, WFT, HADAMARD Transform, DC.

3. Image Enhancement: (by SPATIAL Domain Methods) a Arithmetic and logical operations, pixel er point operations, size operations, b. Smoothing filters-Mean, Medim, Mode filters Comparative study c. Edge enhancement filters-incrocial filters, Sobel, Laplacian, Robert, KIRSCH Homogeneity & DIFF Filters, prewin fiter, Contrast Based edge enhancement techniques. comparative study d. Low Pass filters, High Pass filters, sharpening filters. Comparative Study eComparative studyof all filters £Color image processing.

4. Image Enhancement: (By Frequency Domain Methods) design of Low pass, High pass, EDGE Enhancement, smoothening filters in Frequency Domin. Butter worth filter, Homomorphic filters in Frequency Domain Advantages of filters in frequency domain comparative study of filters in frequency domain and spatial domain.

5. Image Compression: Definition: A brief discussion on-Run length encodingcontour coding Huffman code, compression due to change in demain, compression due to quantization Compression at the time of image transmission. Brief discussion on-Image Compression standards.

6. Image Segmentation: Definition, characteristics of segmentation. Detection of Disocotimaties, Thresholding Pixel based segmentation method. Region based segmentation methods-segmenta-tion by pixel aggregation, segmentation by sub region aggregation, histogram based segmentation, spilt and merge technique. Use of motion in segmentation (spatial domain technique only.

7. Morphology: Dilation, Erosion, Opening, closing Hit-and-Miss transform, Boundary extraction, Region filling, connected components, thinning, Thickening, skeletone, Pruning Extensions to Gray-Scale Images Application of Morphology.

Course objectives

2.11 operating system lab

To gain hands-on experience with operating system concepts and technologies. To learn about different types of operating systems such as Windows, Linux, and macOS. To gain proficiency in using operating system utilities and commands for system administration and troubleshooting.

To learn about process management techniques such as scheduling, synchronization, and deadlock avoidance. To understand the principles of memory management and virtual memory. To gain an understanding of file systems and storage management. To learn about network protocols and the role of the operating system in network communication.

To gain an appreciation for the ethical and social implications of operating systems in fields such as security, privacy, and social justice. Overall, the objective of an Operating System lab is to provide students with hands-on experience with operating system technologies and their role in modern computing, which will prepare them for careers in system administration, software development, and other related fields.

Syllabus:

1. Study of laboratory environment: Hardware specifications, software specifications 2. Simple Unix-C programs: Programs using system calls, library function calls to display and write strings on standard output device and files. 3. Programs using fork system calls. 4. Programs for error reporting using errno, perror () function. 5. Programs using pipes. 6. Shell programming. 7. Programs to simulate process scheduling like FCFS Shortest Job First and Round Robin. 8. Programs to simulate page replacement algorithms like FIFO, Optimal & LRU 9. Programs to simulate free space management. 10. Programs to simulate virtual memory II Programs to simulate deadlock detection

Course objectives

2.12 file structures lab

To gain hands-on experience with file structure concepts and technologies. To learn about different types of file structures such as arrays, linked lists, trees, and graphs. To gain proficiency in implementing and manipulating file structures using programming languages such as C, C++, or Java. To learn about different file organization techniques such as sequential, direct, and indexed file organization.

To understand the principles of data compression and encryption techniques used in file structures. To gain an understanding of file indexing techniques and their role in efficient file searching and retrieval. To learn about database management systems and their relationship to file structures.

To gain an appreciation for the ethical and social implications of file structures in fields such as security, privacy, and social justice. Overall, the objective of a File Structures lab is to provide students with hands-on experience with file structure technologies and their role in modern computing, which will prepare them for careers in database management, software development, and other related fields.

Syllabus:

1. File Operations: Opening reading, writing, closing and creating of files in C++.
2. Study of secondary storage devices: Tracks, sectors, block capacity of disk, tape and CDROMs.
3. File Structures in C++ Reading a stream of fields. record structures and its length indicators, Mixing of numbers and characters. Use of a hex dump, Retrieving records by keys using equential search, direct access
4. File performance: Data compression, storage compacting, reclaiming space dynamically.
5. Indexing and indexed sequential files Index file, inverted file operations, usage of B and B++ trees.
6. Hashing files: Hashing functions, algorithms, record distribution & collision resolution by progressive over flow. Extendable hashing and hashing performance.

Course objectives

2.13 Visual programming lab

To gain hands-on experience with visual programming concepts and technologies. To learn about different visual programming languages such as Scratch, Blockly, and App Inventor. To gain proficiency in designing and developing graphical user interfaces (GUIs) using visual programming tools. To learn about event-driven programming and its applications in visual programming.

To understand the principles of object-oriented programming and their applications in visual programming. To gain an understanding of web development and its relationship to visual programming. To learn about mobile app development and its relationship to visual rogramming. To gain an appreciation for the ethical and social implications of visual programming in fields such as accessibility, usability, and social justice.

Overall, the objective of a Visual Programming lab is to provide students with hands-on experience with visual programming technologies and their role in modern computing, which will prepare them for careers in software development, user experience design, and other related fields.

Syllabus:

Experiments using JAVA AWT/Swing

01 Introduction to AWT & Swings
02 Reading Data from Key Board
03. Handling Buttons
04. Handling with Labels
05. Handling Text Fields.
06. Handling Scroll Bar
07. Handling Text Areas
08. Handling Check Box
09. Handling Radio Buttons
10. Handling Choice Control
11. Handling List Boxes
12. Handling with Menus.
13. Handling Database Using JDBC

Experiments using VC++

1 Reading Data From Key Board
2 Program For Displaying The Cursor While The Data Is Entering..
3. Handling Radio and Check Boxes
4. Handling List box and Combo Box
5. Handling Slider.
6. Handling Menus and Tool Bars
7. Internet Programming
8 creative ActiveX Controls

Course objectives

2.14 DBMS LAB

To gain hands-on experience with DBMS concepts and technologies. To learn about different types of DBMS such as relational, hierarchical, and object-oriented. To gain proficiency in designing and implementing databases using a specific DBMS tool such as Oracle, MySQL, or Microsoft SQL Server.

To learn about SQL (Structured Query Language) and its role in manipulating and querying data. To understand the principles of database normalization and its applications in maintaining data integrity.

To gain an understanding of transaction management and its importance in data consistency and recovery. To learn about database security and privacy issues and their role in modern computing. To gain an appreciation for the ethical and social implications of DBMS in fields such as data governance, social justice, and privacy.

Overall, the objective of a DBMS lab is to provide students with hands-on experience with DBMS technologies and their role in modern computing, which will prepare them for careers in database administration, data analysis, and other related fields.

Syllabus: 1. INTRODUCTION TO RDBMS:

1.1	Advantages of Oracle
1.2	Oracle as a DBA
1.3	Introduction to Oracle tools
1.4	SQL plus
1.5	Data types in Oracle
1.6	DDL commands
1.7	DML commands

2. SQL FOR ORACLE:

2.1	Overview of Data bases
2.2	History of RDBMS
2.3	Providing a given Database in RDBMS
2.4	Under which way RDBMS supports security

3. INTRODUCTION TO SQL:

3.1	Projection
3.2	Restriction
3.3	Different types of Operators
3.4	Between and Operator
3.5	In Operator
3.6	Like Operator
3,7	Is Null Operator
3.8	Cross Product
3.9	Clauses

MCA THIRD YEAR

Course objectives

3.1 Information system control & audit

To gain an understanding of information system control and audit concepts and methodologies. To learn about different types of information systems and their associated risks and controls. To gain proficiency in identifying and assessing information system risks and designing controls to mitigate them. To understand the principles of auditing information systems and evaluating the effectiveness of controls.

To learn about regulatory and compliance requirements related to information systems and how to ensure compliance. To gain an understanding of ethical and social implications of information system control and audit, including issues related to privacy, security, and social justice.

To learn about emerging technologies and their impact on information system control and audit. To gain hands-on experience with information system control and audit tools and techniques.

Overall, the objective of an ISCA course is to provide students with a solid understanding of information system control and audit concepts and methodologies, which will prepare them for careers in information system auditing, risk management, and other related fields.

Syllabus

Introduction: Overview of Information Systems auditing. Conducting an information systems Audit (Chapt1 and 2 of Ron Weber Management Control Framework Top Management Controls (Chapter 3 of Ron Weber) Systems-Development- Management Controls(Chapt4 of Ron Weber)Application Control framework: Boundary Controls, Input Controls, Communication Controls, Processing Controls Database Controls, Output Controls (Chapt10 to 15 of Ron Weber)Generalized Audit Software, Utility Software, Expert Systems, Measures of Asset Safeguarding and Data Integrity, Overview of the Effectiveness of Sys tem Evaluation Process, Evaluation Process of System Efficiency (Chapt. 16, 21,22,23 beginning topics of Ron Weber)

Course objectives

3.2 Network Security

To gain an understanding of the principles and concepts of network security, including confidentiality, integrity, availability, and non-repudiation. To learn about common network security threats and attacks, including malware, phishing, and denial of service (DoS) attacks. To gain proficiency in identifying vulnerabilities in network systems and designing controls to mitigate them.

To understand the principles of cryptography and their applications in securing network communications. To learn about network security protocols such as Secure Sockets Layer (SSL), Transport Layer Security (TLS), and Internet Protocol Security (IPsec).

To gain an understanding of firewall and intrusion detection and prevention systems and their role in network security. To learn about legal and ethical issues related to network security, including data privacy and security breaches. To gain hands-on experience with network security tools and techniques.

Overall, the objective of a Network Security course is to provide students with a solid understanding of network security concepts and methodologies, which will prepare them for careers in network security, cybersecurity, and other related fields.

Syllabus

Introduction: Terminology – notation- primer on networking -types of attacks-Layer and cryptography-authorization-key escrow -Viruses, worms and Trojan Horses-Multi Level mode of Security -legal issues

CRYPTOGRAPHY Introduction-Secret Key cryptography-Public Key Cryptography-Hash algorithm-DES-IDEA AES-Modes of Operations-Hashes and Message Digests-MD2-MD4-MDS SHA-1-RSA-Dime-Hellamn- Digital Signature Standard(DSS)-Elliptic Curve Cryptography.

AUTHENTICATION: Password based authentication-address based authentication-Cryptographic authentication Protocols-Passwords as cryptographic keys-trusted Intermediaries-certificate revocation-Multiple trusted Intermediaries-Session Key Establishment-Delegation.

STANDARDS: KKerberos v4 Kerbos V5- public key infrastructure Real Time communication Security -IP sec: AH and Esp-ipsec:ike-ssl/tls ELECTRONIC MAIL, Mail security-Pem and S/MIME AND PCP

Course objectives

3.3 Object oriented software engineering

To gain an understanding of the principles and concepts of object-oriented programming and design. To learn about software engineering methodologies and practices, including requirements gathering, software design, implementation, testing, and maintenance.

To gain proficiency in object-oriented programming languages such as Java or C++, and related development tools such as Integrated Development Environments (IDEs) and version control systems.

To understand the principles of object-oriented software design patterns and their applications in building modular and extensible software systems. To learn about software testing and quality assurance, including unit testing, integration testing, and user acceptance testing.

To gain an understanding of project management methodologies and their role in software engineering, including Agile and Waterfall methodologies. To learn about legal and ethical issues related to software engineering, including intellectual property and software piracy. To gain hands-on experience with software engineering tools and techniques.

Overall, the objective of an Object-Oriented Software Engineering course is to provide students with a solid understanding of software engineering concepts and methodologies, with a focus on object-oriented programming and design principles, which will prepare them for careers in software development, software engineering, and related fields.

Syllabus:

1. Software Engineering Software related problems, Software Engineering, Concepts, Development activities
2. Modeling: Concepts, Modeling with UML
3. Project Organization & Communication: Project Organization & Communication concepts and their activities
4. Requirements: Requirements elicitation & its activities and managing requirements elicitation
5. Analysis: Analysis overview, concepts, activities and managing analysis
6. System Design: Design overview, concepts, and activities, addressing design goals & managing system design
7. Object Design: Object reuse, its activities & managing reuse, Interface specification concepts & its activities and Managing object design
8. Testing: Testing - concepts, activities & managing testing agement overview, concepts, activities & managing configuration management.

Course objectives

3.4 Embedded Systems

To gain an understanding of the principles and concepts of embedded systems, including hardware and software components, real-time systems, and system-on-chip (SoC) architectures. To learn about programming languages and development tools used in embedded systems, such as C/C++, assembly language, and Integrated Development Environments (IDEs).

To gain proficiency in designing and implementing embedded systems, including designing hardware interfaces, programming microcontrollers, and integrating software and hardware components. To understand the principles of real-time operating systems (RTOS) and their applications in embedded systems. To learn about system-level debugging techniques and tools, including logic analyzers and oscilloscopes.

To gain an understanding of low-level communication protocols, such as Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I2C), and Universal Asynchronous Receiver-Transmitter (UART). To learn about sensor and actuator interfaces and their applications in embedded systems, such as accelerometers, gyroscopes, and motor control systems. To gain hands-on experience with embedded systems development tools and techniques.

Overall, the objective of an Embedded Systems course is to provide students with a solid understanding of embedded systems concepts and methodologies, which will prepare them for careers in embedded systems development, embedded software engineering, and related fields.

Syllabus:

Examples of Embedded systems and Typical hardware Hardware Fundamentals for Software Engineer and Advanced Hardware Fundamentals Interrupts and Survey of software architectures. Introduction to RTOS and More Operating System Services Basic Design using RTOS Embedded Software development tools and Debugging Techniques.

Course objectives

3.5 Data ware housing & Data Mining

To gain an understanding of the principles and concepts of data warehousing, including data modeling, data extraction, data transformation, and data loading. To learn about data mining techniques and algorithms, including association rule mining, classification, clustering, and anomaly detection.

To gain proficiency in using data warehousing and data mining tools, such as Oracle Data Warehouse Builder, Microsoft SQL Server Analysis Services, and IBM Cognos.

To understand the principles of data preprocessing and its importance in data mining, including missing value imputation, outlier detection, and feature selection.

To learn about the applications of data mining in various domains, such as business, healthcare, and social media. To gain an understanding of data privacy and security issues related to data warehousing and data mining, including data anonymization and de-identification.
To learn about legal and ethical issues related to data warehousing and data mining, including data ownership and privacy laws. To gain hands-on experience with data warehousing and data mining tools and techniques.

Overall, the objective of a Data Warehousing and Data Mining course is to provide students with a solid understanding of data warehousing and data mining concepts and methodologies, which will prepare them for careers in data analysis, business intelligence, and related fields.

Syllabus:

1. Introduction to Data Mining: Motivation and importance, What is Data Mining, Relational Databases, Data Warehouses, Transactional Databases, Advanced Database Systems and Advanced Database Applications, Data Mining Functionalities, Interestingness of a pattern Classification of Data Mining Systems, Major issues in Data Mining,

2. Data Warehouse and OLAP Technology for Data Mining What is a Data Warehouse? Multi-Dimensional Data Model, Data Warehouse Architecture, Data Warehouse Implementation, Development of Data Cube Technology, Data Warehousing to Data Mining.

3. Data Preprocessing: Why Pre-process the Data? Data Cleaning, Data Integration and Transformation Data Reduction, Discretization and Concept Hierarchy Generation.

4. Data Mining Primitives, Languages and system Architectures: Data Mining Primitives: What defines a Data Mining Task? A Data Mining query language Designing Graphical Use Interfaces Based on a Data Mining Query language Architectures of Data Mining Systems.

5. Concept Description: Characterization and comparison: What is Concept Description? Data Generalization and summarization-based Characterization, Analytical Characterization: Analysis of Attribute Relevance, Mining Class Comparisons: Discriminating between different Classes, Mining Descriptive Statistical Measures in large Databases.

6. Mining Association rule in large Databases Association Rule Mining, Mining Single - Dimensional Boolean Association Rules from Transactional Databases, Mining Multilevel Association Rules from Transaction Databases, Mining Multidimensional Association Rules from Relational Databases and Data Warehouses, From Association Mining to Correlation Analysis, Constraint-Based Association Mining.

7. Classification and prediction Concepts and Issues regarding Classification and Prediction, Classification by Decision Tree Induction, Bayesian Classification, Classification by Backpropagation, Classification Basedon Concepts from Association Rule Mining,other Classification Methods like k-Nearest Neighbor classifiers, Case-Based Reasoning, Generic Algorithms, Rough Set Approach, Fuzzy Set Approaches, Prediction, Classifier Accuracy.

8. Cluster Analysis: What is Cluster Analysis? Types of Data in Cluster Analysis, A Categorization of Major Clustering Methods Text Book: Data Mining Concepts and Techniques, Jiawei Han and Micheline Kamber, Morgan Kaufman Publications

Course objectives

3.6 OOSE Lab

To gain hands-on experience with object-oriented programming (OOP) concepts and techniques, such as classes, objects, inheritance, and polymorphism. To learn about software design principles and patterns, such as Model-View-Controller (MVC), Singleton, and Observer. To gain proficiency in using software development tools and techniques, such as version control systems (e.g., Git), integrated development environments (IDEs), and testing frameworks.

To understand the principles of software architecture and its importance in large-scale software development projects. To learn about software development methodologies, such as Agile and Waterfall, and their applications in OOSE. To gain an understanding of software requirements gathering and analysis, including use case modeling and user interface design.

To learn about software testing techniques and their importance in ensuring software quality and reliability.To gain hands-on experience with OOSE tools and techniques, such as UML modeling, object-oriented design patterns, and software testing frameworks.

Overall, the objective of an OOSE lab is to provide students with hands-on experience with OOP concepts and software engineering tools and techniques, which will prepare them for careers in software development, software engineering, and related fields.

Syllabus:

The purpose of the Software Engineering Lab course is to familiarize the students with modern software engineering methods and tools, Rational Products. The course is realized as a project-like assignment that can, in principle, by a team of three/four students working full time.

Typically the assignments have been completed during the semester requiring approximately 80-120 hours from each project team. The goal of the Software Engineering Project is to have a walk through from the requirements, design to implementing and testing. An emphasis is put on proper documentation. Extensive hardware expertise is not necessary, so proportionate attention can be given to the design methodology.

Despite its apparent simplicity, the problem allows plenty of alternative solutions and should be a motivating and educating exercise. Demonstration of a properly functioning system and sufficient documentation is proof of a completed assignment Projects. Term projects are projects that a group student or might take through from initial specification to implementation. The project deliverables include: Documentation including oA problem statement Arequirements document § A Requirements Analysis Document.

A System Requirements Specification. § A Software Requirements Specification. A design document A Software Design Description and a System Design Document. A test specification. Manuals/guides for Users and associated help frames o Programmers Administrators (installation instructions). A project plan and schedule setting out milestones, resource usage and estimated costs. A quality plan setting out quality assurance procedures. An implementation.

Course objectives

3.7 Data communication & Networking Lab

To gain practical experience in configuring network devices such as routers and switches. To learn about network protocols, such as TCP/IP, DNS, DHCP, and FTP, and their implementations. To gain an understanding of network topologies, such as LAN, WAN, and MAN, and their applications in different scenarios. To learn about network security techniques such as firewalls, VPNs, and intrusion detection/prevention systems.

To gain experience with network troubleshooting and analysis tools, such as Wireshark, Ping, and Traceroute. To learn about wireless network technologies, such as Wi-Fi and Bluetooth, and their implementation. To gain an understanding of network management techniques, such as SNMP, and their implementation. To gain hands-on experience with network simulation software such as Cisco Packet Tracer and GNS3.

Overall, the objective of a Data Communication & Networking Lab is to provide students with practical experience in network design, implementation, and troubleshooting, which will prepare them for careers in network engineering, network administration, and related fields.

Syllabus:

First Cycle Of Experiments: 1.1 PC-to-PC COMMUNICATIONS UNDER WIN WIN2000's DIRECT CABLE CONNECTION with NULL MODEM a) Using Serial Ports and RS-232C Cable Connection b) Using Parallel Ports and Direct Parallel Cable Connection 121 PC-to-PC COMMUNICATIONS UNDER WIN 98/WIN2000's DIAL-UP NET WORKING with MODEM and 4-LINE EXCHANGE 13 PC-to-PC COMMUNICATIONS UNDER WIN 987 WIN 2000's HYPER TERMINAL with MODEM and 4-LINE EXCHANGE 14 LAN WITH BUS/STAR(Switch or Hub) TOPOLOGY with a minimum of two systems i) Windows Peer-to-Peer Network ji) Windows NT Client-Server Network 1.5LAN WITH BUS STAR(Switch or Hub) TOPOLOGY with a minimum of two systems using NOVELL Netware 1.6 TERMINAL NETWORK WITH UNIX/LINUX SERVER and one or two Terminals using Serial Ports 1.7 TERMINAL NETWORK WITH UNIX/LINUX SERVERS port Terminal Server and one or two terminals