

DATA STRUCTURES

B.Tech. II Year I Sem.

L T P C
3 0 0 3

Prerequisites: Programming for Problem Solving

Course Objectives

- Exploring basic data structures such as stacks and queues.
- Introduces a variety of data structures such as hash tables, search trees, tries, heaps, graphs.
- Introduces sorting and pattern matching algorithms

Course Outcomes

- Ability to select the data structures that efficiently model the information in a problem.
- Ability to assess efficiency trade-offs among different data structure implementations or combinations.
- Implement and know the application of algorithms for sorting and pattern matching.
- Design programs using a variety of data structures, including hash tables, binary and general tree structures, search trees, tries, heaps, graphs, and AVL-trees.

UNIT - I

Introduction to Data Structures, abstract data types, Linear list – singly linked list implementation, insertion, deletion and searching operations on linear list, Stacks- Operations, array and linked representations of stacks, stack applications, Queues- operations, array and linked representations.

UNIT - II

Dictionaries: linear list representation, skip list representation, operations - insertion, deletion and searching.

Hash Table Representation: hash functions, collision resolution-separate chaining, open addressing-linear probing, quadratic probing, double hashing, rehashing, extendible hashing.

UNIT - III

Search Trees: Binary Search Trees, Definition, Implementation, Operations- Searching, Insertion and Deletion, B- Trees, B+ Trees, AVL Trees, Definition, Height of an AVL Tree, Operations – Insertion, Deletion and Searching, Red -Black, Splay Trees.

UNIT - IV

Graphs: Graph Implementation Methods. Graph Traversal Methods.

Sorting: Quick Sort, Heap Sort, External Sorting- Model for external sorting, Merge Sort.

UNIT - V

Pattern Matching and Tries: Pattern matching algorithms-Brute force, the Boyer -Moore algorithm, the Knuth-Morris-Pratt algorithm, Standard Tries, Compressed Tries, Suffix tries.

TEXT BOOKS:

1. Fundamentals of Data Structures in C, 2 nd Edition, E. Horowitz, S. Sahni and Susan Anderson Freed, Universities Press.
2. Data Structures using C - A. S.Tanenbaum, Y. Langsam, and M.J. Augenstein, PHI/Pearson Education.

REFERENCE BOOK:

1. Data Structures: A Pseudocode Approach with C, 2 nd Edition, R. F. Gilberg and B.A.Forouzan, Cengage Learning.

COMPUTER ORGANIZATION AND ARCHITECTURE

B.Tech. II Year I Sem.

L T P C
3 0 0 3

Co-requisite: A Course on "Digital Electronics".

Course Objectives

- The purpose of the course is to introduce principles of computer organization and the basic architectural concepts.
- It begins with basic organization, design, and programming of a simple digital computer and introduces simple register transfer language to specify various computer operations.
- Topics include computer arithmetic, instruction set design, microprogrammed control unit, pipelining and vector processing, memory organization and I/O systems, and multiprocessors

Course Outcomes

- Understand the basics of instruction sets and their impact on processor design.
- Demonstrate an understanding of the design of the functional units of a digital computer system.
- Evaluate cost performance and design trade-offs in designing and constructing a computer processor including memory.
- Design a pipeline for consistent execution of instructions with minimum hazards.
- Recognize and manipulate representations of numbers stored in digital computers

UNIT - I

Digital Computers: Introduction, Block diagram of Digital Computer, Definition of Computer Organization, Computer Design and Computer Architecture.

Register Transfer Language and Micro operations: Register Transfer language, Register Transfer, Bus and memory transfers, Arithmetic Micro operations, logic micro operations, shift micro operations, Arithmetic logic shift unit.

Basic Computer Organization and Design: Instruction codes, Computer Registers Computer instructions, Timing and Control, Instruction cycle, Memory Reference Instructions, Input - Output and Interrupt.

UNIT - II

Microprogrammed Control: Control memory, Address sequencing, micro program example, design of control unit.

Central Processing Unit: General Register Organization, Instruction Formats, Addressing modes, Data Transfer and Manipulation, Program Control.

UNIT - III

Data Representation: Data types, Complements, Fixed Point Representation, Floating Point Representation.

Computer Arithmetic: Addition and subtraction, multiplication Algorithms, Division Algorithms, Floating - point Arithmetic operations. Decimal Arithmetic unit, Decimal Arithmetic operations.

UNIT - IV

Input-Output Organization: Input-Output Interface, Asynchronous data transfer, Modes of Transfer, Priority Interrupt Direct memory Access.

Memory Organization: Memory Hierarchy, Main Memory, Auxiliary memory, Associate Memory, Cache Memory.

UNIT - V

Reduced Instruction Set Computer: CISC Characteristics, RISC Characteristics.

SR23 B.Tech. CSE (AI and ML) Syllabus

Pipeline and Vector Processing: Parallel Processing, Pipelining, Arithmetic Pipeline, Instruction Pipeline, RISC Pipeline, Vector Processing, Array Processor.

Multi Processors: Characteristics of Multiprocessors, Interconnection Structures, Interprocessor arbitration, Interprocessor communication and synchronization, Cache Coherence.

TEXT BOOK:

1. Computer System Architecture - M. Morris Mano, Third Edition, Pearson/PHI.

REFERENCE BOOKS:

1. Computer Organization - Carl Hamacher, Zvonks Vranesic, SafeaZaky, V th Edition, McGraw Hill.
2. Computer Organization and Architecture -William Stallings Sixth Edition, Pearson/PHI.
3. Structured Computer Organization - Andrew S. Tanenbaum, 4 th Edition, PHI/Pearson.



SOFTWARE ENGINEERING

B.Tech. II Year I Sem.

L T P C
3 0 0 3

Course Objectives

- The aim of the course is to provide an understanding of the working knowledge of the techniques for estimation, design, testing and quality management of large software development projects.
- Topics include process models, software requirements, software design, software testing, software process/product metrics, risk management, quality management and UML diagrams

Course Outcomes

- Ability to translate end-user requirements into system and software requirements, using e.g. UML, and structure the requirements in a Software Requirements Document (SRD).
- Identify and apply appropriate software architectures and patterns to carry out high level design of a system and be able to critically compare alternative choices.
- Will have experience and/or awareness of testing problems and will be able to develop a simple testing report

UNIT - I

Introduction to Software Engineering: The evolving role of software, changing nature of software, software myths. **A Generic view of process:** Software engineering- a layered technology, a process framework, the capability maturity model integration (CMMI). **Process models:** The waterfall model, Spiral model and Agile methodology

UNIT - II

Software Requirements: Functional and non-functional requirements, user requirements, system requirements, interface specification, the software requirements document.

Requirements engineering process: Feasibility studies, requirements elicitation and analysis, requirements validation, requirements management.

UNIT - III

Design Engineering: Design process and design quality, design concepts, the design model.

Creating an architectural design: software architecture, data design, architectural styles and patterns, architectural design, conceptual model of UML, basic structural modeling, class diagrams, sequence diagrams, collaboration diagrams, use case diagrams, component diagrams.

UNIT - IV

Testing Strategies: A strategic approach to software testing, test strategies for conventional software, black-box and white-box testing, validation testing, system testing, the art of debugging.

Metrics for Process and Products: Software measurement, metrics for software quality.

UNIT - V

Risk management: Reactive Vs proactive risk strategies, software risks, risk identification, risk projection, risk refinement, RMMM. **Quality Management:** Quality concepts, software quality assurance, software reviews, formal technical reviews, statistical software quality assurance, software reliability, the ISO 9000 quality standards.

TEXT BOOKS:

1. Software Engineering, A practitioner's Approach- Roger S. Pressman, 6th edition, McGraw Hill International Edition.
2. Software Engineering- Sommerville, 7th edition, Pearson Education.

REFERENCE BOOKS:

1. The unified modeling language user guide Grady Booch, James Rumbaugh, Ivar Jacobson, Pearson Education.
2. Software Engineering, an Engineering approach- James F. Peters, Witold Pedrycz, John Wiley.
3. Software Engineering principles and practice- Waman S Jawadekar, The McGraw-Hill Companies.
4. Fundamentals of object-oriented design using UML Meiler page-Jones: Pearson Education.



OPERATING SYSTEMS

B.Tech. II Year I Sem.

L	T	P	C
3	0	0	3

Prerequisites:

1. A course on "Computer Programming and Data Structures".
2. A course on "Computer Organization and Architecture".

Course Objectives:

- Introduce operating system concepts (i.e., processes, threads, scheduling, synchronization, deadlocks, memory management, file and I/O subsystems and protection)
- Introduce the issues to be considered in the design and development of operating system
- Introduce basic Unix commands, system call interface for process management, interprocess communication and I/O in Unix

Course Outcomes:

- Will be able to control access to a computer and the files that may be shared
- Demonstrate the knowledge of the components of computers and their respective roles in computing.
- Ability to recognize and resolve user problems with standard operating environments.
- Gain practical knowledge of how programming languages, operating systems, and architectures interact and how to use each effectively.

UNIT - I

Operating System - Introduction, Structures - Simple Batch, Multiprogrammed, Time-shared, Personal Computer, Parallel, Distributed Systems, Real-Time Systems, System components, Operating System services, System Calls

Process - Process concepts and scheduling, Operations on processes, Cooperating Processes, Threads

UNIT - II

CPU Scheduling - Scheduling Criteria, Scheduling Algorithms, Multiple -Processor Scheduling. System call interface for process management-fork, exit, wait, waitpid, exec

Deadlocks - System Model, Deadlocks Characterization, Methods for Handling Deadlocks, Deadlock Prevention, Deadlock Avoidance, Deadlock Detection, and Recovery from Deadlock

UNIT - III

Process Management and Synchronization - The Critical Section Problem, Synchronization Hardware, Semaphores, and Classical Problems of Synchronization, Critical Regions, Monitors

Interprocess Communication Mechanisms: IPC between processes on a single computer system, IPC between processes on different systems, using pipes, FIFOs, message queues, shared memory.

UNIT - IV

Memory Management and Virtual Memory - Logical versus Physical Address Space, Swapping, Contiguous Allocation, Paging, Segmentation, Segmentation with Paging, Demand Paging, Page Replacement, Page Replacement Algorithms.

UNIT - V

File System Interface and Operations -Access methods, Directory Structure, Protection, File System Structure, Allocation methods, Free-space Management. Usage of open, create, read, write, close, lseek, stat, ioctl system calls.

TEXT BOOKS:

1. Operating System Principles- Abraham Silberchatz, Peter B. Galvin, Greg Gagne 7th Edition, John Wiley.
2. Advanced programming in the UNIX environment, W.R. Stevens, Pearson education.

REFERENCE BOOKS:

1. Operating Systems- Internals and Design Principles, William Stallings, Fifth Edition-2005, Pearson Education/PHI
2. Operating System A Design Approach- Crowley, TMH.
3. Modern Operating Systems, Andrew S. Tanenbaum 2nd edition, Pearson/PHI
4. UNIX programming environment, Kernighan and Pike, PHI/ Pearson Education
5. UNIX Internals -The New Frontiers, U. Vahalia, Pearson Education.



DATA STRUCTURES WITH C LAB

B.Tech. II Year I Sem.

L T P C
0 0 2 1

Prerequisites: A Course on "Programming for problem solving".

Course Objectives:

- It covers various concepts of C programming language
- It introduces searching and sorting algorithms
- It provides an understanding of data structures such as stacks and queues.

Course Outcomes:

- Ability to develop C programs for computing and real-life applications using basic elements like control statements, arrays, functions, pointers and strings, and data structures like stacks, queues and linked lists.
- Ability to Implement searching and sorting algorithms

List of Experiments:

1. Write a program that uses functions to perform the following operations on singly linked list.:
i) Creation ii) Insertion iii) Deletion iv) Traversal
2. Write a program that uses functions to perform the following operations on doubly linked list.:
i) Creation ii) Insertion iii) Deletion iv) Traversal
3. Write a program that uses functions to perform the following operations on circular linked list.:
i) Creation ii) Insertion iii) Deletion iv) Traversal
4. Write a program that implement stack (its operations) using
i) Arrays ii) Pointers
5. Write a program that implement Queue (its operations) using
i) Arrays ii) Pointers
6. Write a program that implements the following sorting methods to sort a given list of integers in ascending order
i) Quick sort ii) Heap sort iii) Merge sort
7. Write a program to implement the tree traversal methods(Recursive and Non Recursive).
8. Write a program to implement
i) Binary Search tree ii) B Trees iii) B+ Trees iv) AVL trees
v) Red - Black trees
9. Write a program to implement the graph traversal methods.
10. Implement a Pattern matching algorithms using Boyer- Moore, Knuth-Morris-Pratt

TEXT BOOKS:

1. Fundamentals of Data Structures in C, 2nd Edition, E. Horowitz, S. Sahni and Susan Anderson Freed, Universities Press.
2. Data Structures using C - A. S. Tanenbaum, Y. Langsam, and M. J. Augenstein, PHI/Pearson Education.

REFERENCE BOOK:

1. Data Structures: A Pseudocode Approach with C, 2nd Edition, R. F. Gilberg and B. A. Forouzan, Cengage Learning.

OPERATING SYSTEMS LAB

B.Tech. II Year I Sem.

L T P C
0 0 2 1

Prerequisites: A course on "Programming for Problem Solving", A course on "Computer Organization and Architecture".

Co-requisite: A course on "Operating Systems".

Course Objectives:

- To provide an understanding of the design aspects of operating system concepts through simulation
- Introduce basic Unix commands, system call interface for process management, interprocess communication and I/O in Unix

Course Outcomes:

- Simulate and implement operating system concepts such as scheduling, deadlock management, file management and memory management.
- Able to implement C programs using Unix system calls

List of Experiments:

1. Write C programs to simulate the following CPU Scheduling algorithms a) FCFS b) SJF c) Round Robin d) priority
2. Write programs using the I/O system calls of UNIX/LINUX operating system (open, read, write, close, fcntl, seek, stat, opendir, readdir)
3. Write a C program to simulate Bankers Algorithm for Deadlock Avoidance and Prevention.
4. Write a C program to implement the Producer – Consumer problem using semaphores using UNIX/LINUX system calls.
5. Write C programs to illustrate the following IPC mechanisms a) Pipes b) FIFOs c) Message Queues d) Shared Memory
6. Write C programs to simulate the following memory management techniques a) Paging b) Segmentation
7. Write C programs to simulate Page replacement policies a) FCFS b) LRU c) Optimal

TEXT BOOKS:

1. Operating System Principles- Abraham Silberchatz, Peter B. Galvin, Greg Gagne 7th Edition, John Wiley
2. Advanced programming in the Unix environment, W.R.Stevens, Pearson education.

REFERENCE BOOKS:

1. Operating Systems – Internals and Design Principles, William Stallings, Fifth Edition-2005, Pearson Education/PHI
2. Operating System - A Design Approach-Crowley, TMH.
3. Modern Operating Systems, Andrew S Tanenbaum, 2nd edition, Pearson/PHI
4. UNIX Programming Environment, Kernighan and Pike, PHI/Pearson Education
5. UNIX Internals: The New Frontiers, U. Vahalia, Pearson Education



SOFTWARE ENGINEERING LAB

B.Tech. II Year I Sem.

L T P C
0 0 2 1

Prerequisites

- A course on "Programming for Problem Solving".

Co-requisite

- A Course on "Software Engineering".

Course Objectives:

- To have hands on experience in developing a software project by using various software engineering principles and methods in each of the phases of software development.

Course Outcomes:

- Ability to translate end-user requirements into system and software requirements
- Ability to generate a high-level design of the system from the software requirements
- Will have experience and/or awareness of testing problems and will be able to develop a simple testing report

List of Experiments

Do the following seven exercises for any two projects given in the list of sample projects or any other Projects:

1. Development of problem statements.
2. Preparation of Software Requirement Specification Document, Design Documents and Testing Phase related documents.
3. Preparation of Software Configuration Management and Risk Management related documents.
4. Study and usage of any Design phase CASE tool
5. Performing the Design by using any Design phase CASE tools.
6. Develop test cases for unit testing and integration testing
7. Develop test cases for various white box and black box testing techniques.

Sample Projects:

1. Passport automation System
2. Book Bank
3. Online Exam Registration
4. Stock Maintenance System
5. Online course reservation system
6. E-ticketing
7. Software Personnel Management System
8. Credit Card Processing
9. E-book management System.
10. Recruitment system

TEXT BOOKS:

1. Software Engineering, A practitioner's Approach- Roger S. Pressman, 6th edition, McGraw Hill International Edition.
2. Software Engineering- Sommerville, 7th edition, Pearson Education.
3. The unified modeling language user guide Grady Booch, James Rumbaugh, Ivar Jacobson, Pearson Education.

REFERENCE BOOKS:

1. Software Engineering, an Engineering approach- James F. Peters, Witold Pedrycz, John Wiley.
2. Software Engineering principles and practice- Waman S Jawadekar, The McGraw-Hill



FULL STACK WEB APP DEVELOPMENT LAB

B.Tech. II Year I Sem.

L T P C
0 0 2 1

Prerequisites: Object Oriented Programming through Java, HTML Basics

Course Objectives:

- To implement the static web pages using HTML and do client side validation using JavaScript.
- To design and work with databases using Java
- To develop an end to end application using java full stack.
- To introduce Node JS implementation for server side programming.
- To experiment with single page application development using React.

Course Outcomes: At the end of the course, the student will be able to,

- Build a custom website with HTML, CSS, and Bootstrap and little JavaScript.
- Demonstrate Advanced features of JavaScript and learn about JDBC
- Develop Server - side implementation using Java technologies like
- Develop the server - side implementation using Node JS.
- Design a Single Page Application using React.

Exercises:

1. Build a responsive web application for shopping cart with registration, login, catalog and cart pages using CSS3 features, flex and grid.
2. Make the above web application responsive web application using Bootstrap framework.
3. Use JavaScript for doing client - side validation of the pages implemented in experiment 1 and experiment 2.
4. Explore the features of ES6 like arrow functions, callbacks, promises, async/await. Implement an application for reading the weather information from openweathermap.org and display the information in the form of a graph on the web page.
5. Develop a java stand alone application that connects with the database (Oracle / mySql) and perform the CRUD operation on the database tables.
6. Create an xml for the bookstore. Validate the same using both DTD and XSD.
7. Design a controller with servlet that provides the interaction with application developed in experiment 1 and the database created in experiment 5.
8. Maintaining the transactional history of any user is very important. Explore the various session tracking mechanism (Cookies, HTTP Session)
9. Create a custom server using http module and explore the other modules of Node JS like OS, path, event.
10. Develop an express web application that can interact with REST API to perform CRUD operations on student data. (Use Postman)
11. For the above application create authorized end points using JWT (JSON Web Token).
12. Create a react application for the student management system having registration, login, contact, about pages and implement routing to navigate through these pages.
13. Create a service in react that fetches the weather information from openweathermap.org and the display the current and historical weather information using graphical representation using chart.js
14. Create a TODO application in react with necessary components and deploy it into github.

REFERENCE BOOKS:

1. Jon Duckett, Beginning HTML, XHTML, CSS, and JavaScript, Wrox Publications, 2010
2. Bryan Basham, Kathy Sierra and Bert Bates, Head First Servlets and JSP, O'Reilly Media, 2nd Edition, 2008.
3. Vasan Subramanian, Pro MERN Stack, Full Stack Web App Development with Mongo, Express, React, and Node, 2nd Edition, A Press.

AUTOMATA THEORY AND COMPILER DESIGN

B.Tech. II Year II Sem.

L T P C
3 0 0 3

Prerequisite: Nil

Course Objectives

- To introduce the fundamental concepts of formal languages, grammars and automata theory.
- To understand deterministic and non-deterministic machines and the differences between decidability and undecidability.
- Introduce the major concepts of language translation and compiler design and impart the knowledge of practical skills necessary for constructing a compiler.
- Topics include phases of compiler, parsing, syntax directed translation, type checking use of symbol tables, intermediate code generation

Course Outcomes

- Able to employ finite state machines for modeling and solving computing problems.
- Able to design context free grammars for formal languages.
- Able to distinguish between decidability and undecidability.
- Demonstrate the knowledge of patterns, tokens & regular expressions for lexical analysis.
- Acquire skills in using lex tool and design LR parsers

UNIT - I

Introduction to Finite Automata: Structural Representations, Automata and Complexity, the Central Concepts of Automata Theory – Alphabets, Strings, Languages, Problems.

Nondeterministic Finite Automata: Formal Definition, an application, Text Search, Finite Automata with Epsilon-Transitions.

Deterministic Finite Automata: Definition of DFA, How A DFA Process Strings, The language of DFA, Conversion of NFA with ϵ -transitions to NFA without ϵ -transitions. Conversion of NFA to DFA

UNIT - II

Regular Expressions: Finite Automata and Regular Expressions, Applications of Regular Expressions, Algebraic Laws for Regular Expressions, Conversion of Finite Automata to Regular Expressions.

Pumping Lemma for Regular Languages: Statement of the pumping lemma, Applications of the Pumping Lemma.

Context-Free Grammars: Definition of Context-Free Grammars, Derivations Using a Grammar, Leftmost and Rightmost Derivations, the Language of a Grammar, Parse Trees, Ambiguity in Grammars and Languages.

UNIT - III

Push Down Automata: Definition of the Pushdown Automaton, the Languages of a PDA, Equivalence of PDA and CFG's, Acceptance by final state

Turing Machines: Introduction to Turing Machine, Formal Description, Instantaneous description, The language of a Turing machine

Undecidability: Undecidability, A Language that is Not Recursively Enumerable, An Undecidable Problem That is RE, Undecidable Problems about Turing Machines

UNIT - IV

Introduction: The structure of a compiler,

Lexical Analysis: The Role of the Lexical Analyzer, Input Buffering, Recognition of Tokens, The Lexical- Analyzer Generator Lex,

Syntax Analysis: Introduction, Context-Free Grammars, Writing a Grammar, Top-Down Parsing, Bottom- Up Parsing, Introduction to LR Parsing: Simple LR, More Powerful LR Parsers

UNIT - V

Syntax-Directed Translation: Syntax-Directed Definitions, Evaluation Orders for SDD's, Syntax-Directed Translation Schemes, Implementing L-Attributed SDD's.

Intermediate-Code Generation: Variants of Syntax Trees, Three-Address Code

Run-Time Environments: Stack Allocation of Space, Access to Nonlocal Data on the Stack, Heap Management

TEXT BOOKS:

1. Introduction to Automata Theory, Languages, and Computation, 3rd Edition, John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman, Pearson Education.
2. Theory of Computer Science – Automata languages and computation, Mishra and Chandrashekar, 2nd Edition, PHI.

REFERENCE BOOKS:

1. Compilers: Principles, Techniques and Tools, Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, 2nd Edition, Pearson.
2. Introduction to Formal languages Automata Theory and Computation, Kamala Krithivasan, Rama R, Pearson.
3. Introduction to Languages and The Theory of Computation, John C Martin, TMH.
4. lex & yacc - John R. Levine, Tony Mason, Doug Brown, O'reilly Compiler Construction, Kenneth C. Loudon, Thomson. Course Technology.



DATABASE MANAGEMENT SYSTEMS

B.Tech. II Year II Sem.

L T P C
3 0 0 3

Prerequisites: A course on "Data Structures".

Course Objectives:

- To understand the basic concepts and the applications of database systems.
- To master the basics of SQL and construct queries using SQL.
- Topics include data models, database design, relational model, relational algebra, transaction control, concurrency control, storage structures and access techniques.

Course Outcomes:

- Gain knowledge of fundamentals of DBMS, database design and normal forms
- Master the basics of SQL for retrieval and management of data.
- Be acquainted with the basics of transaction processing and concurrency control.
- Familiarity with database storage structures and access techniques

UNIT - I

Database System Applications: A Historical Perspective, File Systems versus a DBMS, the Data Model, Levels of Abstraction in a DBMS, Data Independence, Structure of a DBMS

Introduction to Database Design: Database Design and ER Diagrams, Entities, Attributes, and Entity Sets, Relationships and Relationship Sets, Additional Features of the ER Model, Conceptual Design With the ER Model

UNIT - II

Introduction to the Relational Model: Integrity constraint over relations, enforcing integrity constraints, querying relational data, logical database design, introduction to views, destroying/altering tables and views.

Relational Algebra, Tuple relational Calculus, Domain relational calculus.

UNIT - III

SQL: QUERIES, CONSTRAINTS, TRIGGERS: form of basic SQL query, UNION, INTERSECT, and EXCEPT, Nested Queries, aggregation operators, NULL values, complex integrity constraints in SQL, triggers and active databases.

Schema Refinement: Problems caused by redundancy, decompositions, problems related to decomposition, reasoning about functional dependencies, First, Second, Third normal forms, BCNF, lossless join decomposition, multivalued dependencies, Fourth normal form, Fifth normal form.

UNIT - IV

Transaction Concept, Transaction State, Implementation of Atomicity and Durability, Concurrent Executions, Serializability, Recoverability, Implementation of Isolation, Testing for serializability, Lock Based Protocols, Timestamp Based Protocols, Validation- Based Protocols, Multiple Granularity, Recovery and Atomicity, Log-Based Recovery, Recovery with Concurrent Transactions.

UNIT - V

Data on External Storage, File Organization and Indexing, Cluster Indexes, Primary and Secondary Indexes, Index data Structures, Hash Based Indexing, Tree based Indexing, Comparison of File Organizations, Indexes- Intuitions for tree Indexes, Indexed Sequential Access Methods (ISAM), B+ Trees: A Dynamic Index Structure.

TEXT BOOKS:

1. Database System Concepts, Silberschatz, Korth, McGraw hill, V edition.3rd Edition

2. Database Management Systems, Raghurama Krishnan, Johannes Gehrke, Tata Mc Graw Hill

REFERENCE BOOKS:

1. Database Systems design, Implementation, and Management, Peter Rob & Carlos Coronel 7th Edition.
2. Fundamentals of Database Systems, Elmasri Navrate, Pearson Education
3. Introduction to Database Systems, C. J. Date, Pearson Education
4. Oracle for Professionals, The X Team, S.Shah and V. Shah, SPD.
5. Database Systems Using Oracle: A Simplified guide to SQL and PL/SQL, Shah, PHI.
6. Fundamentals of Database Management Systems, M. L. Gillenson, Wiley Student Edition.



INTRODUCTION TO ARTIFICIAL INTELLIGENCE

B.Tech. II Year II Sem.

L T P C
3 0 0 3

Prerequisite: Knowledge on Data Structures.

Course Objectives:

- To learn the distinction between optimal reasoning Vs. human like reasoning.
- To understand the concepts of state space representation, exhaustive search, heuristic search together with the time and space complexities.
- To learn different knowledge representation techniques.
- To understand the applications of AI, namely game playing, theorem proving, and machine learning.

Course Outcomes:

- Learn the distinction between optimal reasoning Vs human like reasoning and formulate an efficient problem space for a problem expressed in natural language. Also select a search algorithm for a problem and estimate its time and space complexities.
- Apply AI techniques to solve problems of game playing, theorem proving, and machine learning.
- Learn different knowledge representation techniques.
- Understand the concepts of state space representation, exhaustive search, heuristic search together with the time and space complexities.
- Comprehend the applications of Probabilistic Reasoning and Bayesian Networks.
- Analyze Supervised Learning Vs. Learning Decision Trees

UNIT - I

Introduction to AI - Intelligent Agents, Problem-Solving Agents,

Searching for Solutions - Breadth-first search, Depth-first search, Hill-climbing search, Simulated annealing search, Local Search in Continuous Spaces.

UNIT-II

Games - Optimal Decisions in Games, Alpha-Beta Pruning, Defining Constraint Satisfaction Problems, Constraint Propagation, Backtracking Search for CSPs, Knowledge-Based Agents, **Logic**-Propositional Logic, Propositional Theorem Proving: Inference and proofs, Proof by resolution, Horn clauses and definite clauses.

UNIT-III

First-Order Logic - Syntax and Semantics of First-Order Logic, Using First Order Logic, Knowledge Engineering in First-Order Logic. Inference in First-Order Logic: Propositional vs. First-Order Inference, Unification, Forward Chaining, Backward Chaining, Resolution.

Knowledge Representation: Ontological Engineering, Categories and Objects, Events.

UNIT-IV

Planning - Definition of Classical Planning, Algorithms for Planning with State Space Search, Planning Graphs, other Classical Planning Approaches, Analysis of Planning approaches. Hierarchical Planning.

UNIT-V

Probabilistic Reasoning:

Acting under Uncertainty, Basic Probability Notation Bayes' Rule and Its Use, Probabilistic Reasoning, Representing Knowledge in an Uncertain Domain, The Semantics of Bayesian Networks, Efficient

SR23 B.Tech. CSE (AI and ML) Syllabus

Representation of Conditional Distributions, Approximate Inference in Bayesian Networks, Relational and First- Order Probability.

TEXT BOOK:

1. Artificial Intelligence: A Modern Approach, Third Edition, Stuart Russell and Peter Norvig, Pearson Education.

REFERENCE BOOKS:

1. Artificial Intelligence, 3rd Edn., E. Rich and K. Knight (TMH)
2. Artificial Intelligence, 3rd Edn., Patrick Henry Winston, Pearson Education.
3. Artificial Intelligence, Shivani Goel, Pearson Education.
4. Artificial Intelligence and Expert systems - Patterson, Pearson Education.



OBJECT ORIENTED PROGRAMMING THROUGH JAVA

B.Tech. II Year II Sem.

L T P C
3 0 0 3

Course Objectives

- To Understand the basic object-oriented programming concepts and apply them in problem solving.
- To Illustrate inheritance concepts for reusing the program.
- To Demonstrate multitasking by using multiple threads and event handling
- To Develop data-centric applications using JDBC.
- To Understand the basics of java console and GUI based programming

Course Outcomes

- Demonstrate the behavior of programs involving the basic programming constructs like control structures, constructors, string handling and garbage collection.
- Demonstrate the implementation of inheritance (multilevel, hierarchical and multiple) by using extend and implement keywords
- Use multithreading concepts to develop inter process communication.
- Understand the process of graphical user interface design and implementation using AWT or swings.
- Develop applets that interact abundantly with the client environment and deploy on the server.

UNIT - I

Object oriented thinking and Java Basics- Need for oop paradigm, summary of oop concepts, coping with complexity, abstraction mechanisms. A way of viewing world – Agents, responsibility, messages, methods, History of Java, Java buzzwords, data types, variables, scope and lifetime of variables, arrays, operators, expressions, control statements, type conversion and casting, simple java program, concepts of classes, objects, constructors, methods, access control, this keyword, garbage collection, overloading methods and constructors, method binding, inheritance, overriding and exceptions, parameter passing, recursion, nested and inner classes, exploring string class.

UNIT - II

Inheritance, Packages and Interfaces – Hierarchical abstractions, Base class object, subclass, subtype, substitutability, forms of inheritance specialization, specification, construction, extension, limitation, combination, benefits of inheritance, costs of inheritance. Member access rules, super uses, using final with inheritance, polymorphism- method overriding, abstract classes, the Object class. Defining, Creating and Accessing a Package, Understanding CLASSPATH, importing packages, differences between classes and interfaces, defining an interface, implementing interface, applying interfaces, variables in interface and extending interfaces. Exploring java.io.

UNIT - III

Exception handling and Multithreading-- Concepts of exception handling, benefits of exception handling, Termination or resumptive models, exception hierarchy, usage of try, catch, throw, throws and finally, built in exceptions, creating own exception subclasses. String handling, Exploring java.util. Differences between multithreading and multitasking, thread life cycle, creating threads, thread priorities, synchronizing threads, inter thread communication, thread groups, daemon threads. Enumerations, autoboxing, annotations, generics.

UNIT - IV

Event Handling: Events, Event sources, Event classes, Event Listeners, Delegation event model, handling mouse and keyboard events, Adapter classes. The AWT class hierarchy, user interface components- labels, button, canvas, scrollbars, text components, check box, checkbox groups, choices,

DATABASE MANAGEMENT SYSTEMS LAB

B.Tech. II Year II Sem.

L T P C
0 0 2 1

Co-requisites: "Database Management Systems"

Course Objectives:

- Introduce ER data model, database design and normalization
- Learn SQL basics for data definition and data manipulation

Course Outcomes:

- Design database schema for a given application and apply normalization
- Acquire skills in using SQL commands for data definition and data manipulation.
- Develop solutions for database applications using procedures, cursors and triggers

List of Experiments:

1. Concept design with E-R Model
2. Relational Model
3. Normalization
4. Practicing DDL commands
5. Practicing DML commands
6. A. Querying (using ANY, ALL, UNION, INTERSECT, JOIN, Constraints etc.)
B. Nested, Correlated subqueries
7. Queries using Aggregate functions, GROUP BY, HAVING and Creation and dropping of Views.
8. Triggers (Creation of insert trigger, delete trigger, update trigger)
9. Procedures
10. Usage of Cursors

TEXT BOOKS:

1. Database Management Systems, Raghurama Krishnan, Johannes Gehrke, Tata Mc Graw Hill, 3rd Edition
2. Database System Concepts, Silberschatz, Korth, McGraw Hill, V edition.

REFERENCE BOOKS:

1. Database Systems design, Implementation, and Management, Peter Rob & Carlos Coronel 7th Edition.
2. Fundamentals of Database Systems, Elmasri Navrate, Pearson Education
3. Introduction to Database Systems, C.J. Date, Pearson Education
4. Oracle for Professionals, The X Team, S. Shah and V. Shah, SPD.
5. Database Systems Using Oracle: A Simplified guide to SQL and PL/SQL, Shah, PHI.
6. Fundamentals of Database Management Systems, M. L. Gillenson, Wiley Student Edition.



lists panels - scrollpane, dialogs, menubar, graphics, layout manager - layout manager types - border, grid, flow, card and grid bag.

UNIT - V

Applets – Concepts of Applets, differences between applets and applications, life cycle of an applet, types of applets, creating applets, passing parameters to applets. Swing – Introduction, limitations of AWT, MVC architecture, components, containers, exploring swing- JApplet, JFrame and JComponent, Icons and Labels, text fields, buttons - The JButton class, Check boxes, Radio buttons, Combo boxes, Tabbed Panes, Scroll Panes, Trees, and Tables.

TEXT BOOKS:

1. Java the complete reference, 7th edition, Herbert schildt, TMH.
2. Understanding OOP with Java, updated edition, T. Budd, Pearson education.

REFERENCE BOOKS:

1. An Introduction to programming and OO design using Java, J.Nino and F.A. Hosch, John wiley & sons.
2. An Introduction to OOP, third edition, T. Budd, Pearson education.
3. Introduction to Java programming, Y. Daniel Liang, Pearson education.
4. An introduction to Java programming and object-oriented application development, R.A. Johnson- Thomson.
5. Core Java 2, Vol 1, Fundamentals, Cay.S. Horstmann and Gary Cornell, eighth Edition, Pearson Education.
6. Core Java 2, Vol 2, Advanced Features, Cay.S. Horstmann and Gary Cornell, eighth Edition, Pearson Education
7. Object Oriented Programming with Java, R.Buyya, S.T.Selvi, X.Chu, TMH.
8. Java and Object Orientation, an introduction, John Hunt, second edition, Springer. 9. Maurach's Beginning Java2 JDK 5, SPD.



JAVA PROGRAMMING LAB

B.Tech. II Year II Sem.

L T P C
0 0 2 1

Course Objectives:

- To understand OOP principles.
- To understand the Exception Handling mechanism.
- To understand Java collection framework.
- To understand multithreaded programming.
- To understand swing controls in Java.

Course Outcomes:

- Able to write the programs for solving real world problems using Java OOP principles.
- Able to write programs using Exceptional Handling approach.
- Able to write multithreaded applications.
- Able to write GUI programs using swing controls in Java.

List of Experiments:

1. Use Eclipse or Net bean platform and acquaint yourself with the various menus. Create a test project, add a test class, and run it. See how you can use auto suggestions, auto fill. Try code formatter and code refactoring like renaming variables, methods, and classes. Try debug step by step with a small program of about 10 to 15 lines which contains at least one if else condition and a for loop.
2. Write a Java program to demonstrate the OOP principles. [i.e., Encapsulation, Inheritance, Polymorphism and Abstraction]
3. Write a Java program to handle checked and unchecked exceptions. Also, demonstrate the usage of custom exceptions in real time scenario.
4. Write a Java program on Random Access File class to perform different read and write operations.
5. Write a Java program to demonstrate the working of different collection classes. [Use package structure to store multiple classes].
6. Write a program to synchronize the threads acting on the same object. [Consider the example of any reservations like railway, bus, movie ticket booking, etc.]
7. Write a program to perform CRUD operations on the student table in a database using JDBC.
8. Write a Java program that works as a simple calculator. Use a grid layout to arrange buttons for the digits and for the +, -, *, % operations. Add a text field to display the result. Handle any possible exceptions like divided by zero.
9. Write a Java program that handles all mouse events and shows the event name at the center of the window when a mouse event is fired. [Use Adapter classes]

REFERENCE BOOKS:

1. Java for Programmers, P. J. Deitel and H. M. Deitel, 10th Edition Pearson education.
2. Thinking in Java, Bruce Eckel, Pearson Education.
3. Java Programming, D. S. Malik and P. S. Nair, Cengage Learning.
4. Core Java, Volume 1, 9th edition, Cay S. Horstmann and G Cornell, Pearson.



PROLOG/ LISP/ PYSWIP

B.Tech. II Year II Sem.

L T P C
0 0 2 1

List of Programs:

1. Write simple fact for following:
 - A. Ram likes mango.
 - B. Seema is a girl.
 - C. Bill likes Cindy.
 - D. Rose is red.
 - E. John owns gold
2. Write predicates one converts centigrade temperatures to Fahrenheit, the other checks if a temperature is below freezing.
3. Write a program to solve the Monkey Banana problem
4. WAP in turbo prolog for medical diagnosis and show the advantages and disadvantages of green and red cuts.
5. Write a program to solve the 4-Queen problem.
6. Write a program to solve traveling salesman problems.
7. Write a program to solve water jug problems using Prolog.
8. Write simple Prolog functions such as the following. Take into account lists which are too short.
-- remove the Nth item from the list. -- insert as the Nth item.
9. Assume the prolog predicate `gt(A, B)` is true when A is greater than B. Use this predicate to define the predicate `addLeaf(Tree, X, NewTree)` which is true if `NewTree` is the `Tree` produced by adding the item `X` in a leaf node. `Tree` and `NewTree` are binary search trees. The empty tree is represented by the atom `nil`.
10. Write a Prolog predicate, `countLists(Alist, Ne, NI)`, using accumulators, that is true when `NI` is the number of items that are listed at the top level of `Alist` and `Ne` is the number of empty lists. Suggestion: First try to count the lists, or empty lists, then modify by adding the other counter.
11. Define a predicate `memCount(AList,Blist,Count)` that is true if `Alist` occurs `Count` times within `Blist`. Define without using an accumulator. Use "not" as defined in `utilities.pro`, to make similar cases are unique, or else you may get more than one count as an answer.

Examples:

```
memCount(a,[b,a],N).
```

```
N = 1 ;
```

```
no
```

```
memCount(a,[b,[a,a,[a],c],a],N).
```

```
N = 4 ;
```

```
no
```

```
memCount([a],[b,[a,a,[a],c],a],N).
```

```
N = 1 ;
```

```
No
```

REFERENCE BOOK:

1. PROLOG: Programming for Artificial Intelligence, 3e, by BRATKO, WILEY