MASTER OF COMPUTER APPLICATIONS (MCA)
WITH SPECILIZATION IN ARTIFICIAL INTELLIGENCE
2 YEAR PROGRAMME
2021-23

SEMESTER I

S.No.	Course Code	Course Name	L	T	P	Credits	Types
1	MAI101	Computer Fundamental and C Programming	3	0	0	3	CC1 / FC
2	MAI103	Software Engineering	3	0	0	3	CC2
3	MAI105	Artificial Intelligence	3	0	0	3	CC3 / FC
4	MAI107	Discrete Mathematics	3	0	0	3	CC4
5	MAI109	Theory of Automata	3	0	0	3	CC5
6	MAI111	Operating System	3	0	0	3	CC6
7	MAI113	Data Base Management System	3	0	0	3	CC7
8	ES401	Fundamental of Environmental Science	3	0	0	3	OE1 / AECC
9	MAI181	C Programming Lab	0	0	3	2	CC-L1 / SEC
10	MAI183	Data Base Management System Lab	0	0	3	2	CC-L2 / SEC
11	GP	General Proficiency	Non Credit				
		Total Hours and Credits	**24**	**0**	**6**	**28**	

Computer Fundamental and C Programming

Computer Fundamental and C Programming			
Course Code:	**MAI 101**	**Credits:**	**3**
No. of Lectures (Hrs/Week):	**3**	**Mid Sem Exam Hours:**	**2**
Total No. of Lectures:	**45**	**End Sem Exam Hours:**	**3**

Course Outcomes: By the end of the course the students will be able to:

- Understand the Computer fundamentals.
- Use of various problem solving techniques.
- Understand the C programming fundamentals.
- Understand C by using arrays, functions, structures and union.
- Develop the Programs in C using its advance features.

UNIT-I Computer Fundamentals: Concept of data and information; Components of Computer: Hardware Input Device, Output Device. CPU: Components of CPU; Memory and Storage Devices; Computer Software: System Software and Application Software; Functions of Operating System. Programming Languages: Machine, Assembly, High Level Language, 4GL; Language Translator; Linker, Loader; Classification of Computers: Micro, Mini, Mainframe, Super computer. Advantages of Computer, Limitations of Computer, Range of Applications of Computer, Social concerns of Computer Technology: Positive and Negative Impacts, Computer Crimes, Viruses and their remedial solutions.

UNIT-II Problem Solving: Problem Identification, Analysis, Flowcharts, Decision Tables, Pseudo codes and algorithms, Program Coding, Program Testing and Execution. C Programming Fundamentals: Keywords, Variables and Constants, Structure of a C program. Operators & Expressions: Arithmetic, Unary, Logical, Bit-wise, Assignment & Conditional Operators, Library Functions, Control Statements: Looping using while, do…while, for statements, Nested loops; decision making using if…else, Else If Ladder; Switch, break, Continue and Goto statements.

UNIT-III Arrays & Functions: Declaration and Initialization; Multidimensional Arrays. String: Operations of Strings; Functions: Defining & Accessing User defined functions, Function Prototype, Passing Arguments, Passing array as argument, Recursion, Use of Library Functions; Macro vs. Functions. Pointers: Declarations, Operations on Pointers, Passing to a function, Pointers & Arrays, Array of Pointers, Array accessing through pointers, Pointer to functions, Function returning pointers, Dynamic Memory Allocations.

UNIT-IV Structures and Union: Defining and Initializing Structure, Array within Structure, Array of Structure, Nesting of Structure, Pointer to Structure, Passing structure and its pointer to Functions; Unions: Introduction to Unions and its Utilities. Files Handing: Opening and closing file in C; Create, Read and Write data to a file; Modes of Files, Operations on file using C Library Functions; Working with Command Line Arguments. Program Debugging and types of errors

Reference :

1. Fundamentals of Computers, V. Rajaraman.

2. Computer Concepts and C Programming, P.B. Kotur

3. Let us C ,YashwanthKanetkar

4. ANSI C, Balagurusamy

SOFTWARE ENGINEERING

SOFTWARE ENGINEERING			
Course Code:	**MAI 103**	**Credits:**	**3**
No. of Lectures (Hrs/Week):	**3**	**Mid Sem Exam Hours:**	**2**
Total No. of Lectures:	**45**	**End Sem Exam Hours:**	**3**

COURSE OBJECTIVES

1 Knowledge of basic SW engineering methods and practices and application.
2 A general understanding of software process models.
3 Understanding of software requirements and the SRS documents.
4 Understanding of software design process.
5 Understanding of software coding, testing and maintenance.

COURSE OUTCOME

At the end of the course the students should be able to:
1 Basic knowledge and understanding of the analysis and design of complex systems.
2 Ability to apply software engineering principles and techniques.
3 Ability to develop, maintain and evaluate large-scale software systems.
4 To produce efficient, reliable, robust and cost-effective software solutions.
5 Ability to perform independent research and analysis.

UNIT I SOFTWARE ENGINEERING

Introduction to software engineering: definitions, role of software engineering, planning a software project, defining the problem, developing a solution strategy, planning the development process, software engineering process paradigms, principles of software engineering, software engineering activities.

UNIT II SOFTWARE LIFE CYCLE MODELS

Software Development Life Cycle (SDLC), SDLC models, waterfall model and its variations, prototype model, iterative enhancement model, spiral model, RAD model, comparison of these models, software development teams, software development environments, validation and traceability, maintenance, prototyping requirements, Software project management.

UNIT III REQUIREMENT ANALYSIS AND DESIGN

Software Requirement Specification (SRS): Introduction, need of SRS, significance, characteristics of SRS, Structure of SRS, IEEE standards for SRS design, functional and non-functional requirements, Requirement gathering and analysis, requirement engineering and management.

UNIT IV SOFTWARE DESIGN PROCESS

Software Design: Introduction, design process activities: architectural design, Abstract specification, Interface design, component design, data structure design, algorithm design modular approach, top-down design, bottom-up design, design

methods: data-flow model: data flow diagram, entity-relation-attribute model: E-R diagram, structural model: structure charts, context diagrams, object models: use case modeling, use case diagrams, sequence diagrams, cohesion and coupling.

UNIT V SOFTWARE CODING, TESTING AND MAINTENANCE

Coding, Testing Methods: unit testing, integration testing, system testing, acceptance testing, testing techniques: white box testing, black box testing, thread testing, regression testing, alpha testing, beta testing, static testing, dynamic testing, Evolution of software products, economics of maintenance, category of software maintenance, Role of product development life cycle, deployment model, adaptive maintenance, corrective maintenance, perfective maintenance, enhancement request, proactive defect prevention, problem reporting, problem resolution, software maintenance from customers' perspective, maintenance standard: IEEE-1219, ISO-12207.

Reference Books:

1. Pankaj Jalote, An Integrated Approach to Software Engineering, Narosa Publishing House, New Delhi 1997.
2. Ian Sommerville, Software Engineering, Pearson Education, 2009.
3. Pressman Roger S., Software Engineering: Practitioner's Approach, McGraw-Hill Inc., 2004.
4. Software Engineering: Software Reliability, Testing and Quality Assurance, Nasib S. Gill, Khanna Book Publishing Co (P) Ltd., New Delhi, 2002.

ARTIFICIAL INTELLIGENCE

Artificial Intelligence			
Course Code:	**MAI 105**	**Credits:**	**3**
No. of Lectures (Hrs/Week):	**3**	**Mid Sem Exam Hours:**	**2**
Total No. of Lectures:	**45**	**End Sem Exam Hours:**	**3**

Course Objectives:

- Gain a historical perspective of AI and its foundations.
- Become familiar with basic principles of AI toward problem-solving, inference, perception, knowledge representation, and learning.
- Investigate applications of AI techniques in intelligent agents, expert systems, and machine learning models.
- Experience AI development tools such as an 'AI language', expert system shell, and/or data mining tool.
- Explore the current scope, potential, limitations, and implications of intelligent systems.

Course Outcomes(COs):

At the end of the course, the student will be able to

- Demonstrate knowledge of the building blocks of AI as presented in terms of intelligent agents
- Analyze and formalize the problem as a state space, graph, design heuristics, and select different search or game-based techniques to solve them.
- Develop intelligent algorithms for constraint satisfaction problems and also design intelligent .systems for Game Playing

- Attain the capability to represent various real-life problem domains using logic-based techniques and use this to perform inference or planning.
- Solve reasoning problems with Expert Systems.

UNIT I INTRODUCTION TO ARTIFICIAL INTELLIGENCE

Basic concept of artificial intelligence (AI), history of AI, AI and consciousness, weak and strong AI, physical symbol system hypothesis, comparison of computer and human skills, practical systems based on AI, development of logic, components of AI, Turing Test in AI, Advantages and Disadvantages of AI, Intelligence, Intelligent System, Role of IS, Comparison of various IS, Mind-Body Problem in AI, Chinese Room Experiment in AI, Parallel and Distributed AI.

UNIT II PROBLEM SOLVING THROUGH AI

Defining the problem as state-space search, analyzing the problem, representing the problems from AI viewpoint, production system, developing production rules, characteristics of the production system, algorithm for problem-solving using AI technique.

UNIT III SEARCH TECHNIQUES

Use of search in AI problem solution, blind search techniques, heuristic search techniques, concept of heuristic knowledge, designing of the heuristic function, types of heuristic search techniques: generate and test, best first search, problem reduction using AND-OR graph, local search technique, branch and bound search, memory bounded search technique, local beam search, properties of heuristic search techniques, overestimation and underestimation of heuristic function hill climbing search, simulated annealing search, constraint satisfaction means ends analysis, Tic-Tac Toe Problem, Water Jug problem, Chess Problem, Tower of Hanoi problem, Travelling Salesman problem, Monkey and Banana Problem, Magic Square.

UNIT IV INTRODUCTION TO LOGIC

Introduction, proposition calculus, syntax o propositional calculus, semantics of propositional calculus, well-formed formula, properties of statements, inferencing of propositional logic, predicate logic, syntax of predicate logic, semantics of predicate logic, concept of resolution, resolution algorithm, skolemization, types of resolution unit resolution, binary resolution.

UNIT V AI Techniques and applications

Introduction to Machine Learning, Introduction to Deep Learning, Introduction to Expert system: Introduction phases in building expert systems, Expert system versus traditional systems, rule-based expert systems, blackboard systems, application of expert systems, list of shells and tools, Introduction to Natural Language Processing, AI in future, AI in social Media, AI in Entertainment and education, AI in drones, AI in Automated Computer support, AI in personalized shopping experience, AI in Finance, AI in smart Cars, AI in travel and navigation, AI in smart home devices, AI in security and surveillance, Ai in education, AI in health care, AI in E commerce.

Text Books:

1. Artificial Intelligence, Elanie Reich: Tata mcgraw Hill publishing house, 2008.
2. Artificial Intelligence, Peterson, TataMcGraw Hill, 2008.
3. Artificial Intelligence, Russel and Norvig, Pearson Printice Hall Publication, 2006.
4. Artificial Intelligence, Winston, PHI publication, 2006

Discrete Mathematics

Course Code:	**MAI 107**	**Credits:**	**3**
No. of Lectures (Hrs/Week):	**3**	**Mid Sem Exam Hours:**	**2**
Total No. of Lectures:	**45**	**End Sem Exam Hours:**	**3**

COURSE OBJECTIVE

The course will develop understanding and knowledge of ordered sets, Recurrence relations, Propositions, Logical operators and Lattices.

UNIT – I Definition, examples and basic properties of ordered sets, maps between ordered sets, Mappings, Composition of Mappings, Countability of sets, Partially ordered sets, Hasse diagram, Isomorphic ordered sets, Principle of Mathematical Induction.

UNIT – II Recurrence Relations, Explicit formula for a sequence, solution of Recurrence Relations, Homogeneous Recurrence Relations with constant coefficients, Particular solution of a Difference Equation, Recurssive functions, generating functions, Convolution of Numeric functions, Solution of Recurrence Relation by the method of Generating functions.

UNIT – III Propositions, Basic Logical Operations, Logical Equivalence involving Tautologies and Contradictions, conditional propositions, Quantifier.

UNIT – IV Lattice, Properties of lattice, lattice as algebraic system, lattice isomorphism, Bounded, complemented and distributive lattice.

COURSE OUTCOMES

Students will have the knowledge of

- Ordered sets and Partial Ordered Sets
- Recurrence relations Numeric Functions.
- Proposiions and Quantifiers.
- Lattices and properties of lattices.

REFERENCE BOOKS

1. B A. Davey and H. A. Priestley, Introduction to Lattices and Order, Cambridge University Press, Cambridge, 1990.
2. Edgar G. Goodaire and Michael M. Parmenter, Discrete Mathematics with Graph Theory (2nd Edition), Pearson Education (Singapore) Pte. Ltd., Indian Reprint 2003.
3. Rudolf Lidl and Günter Pilz, Applied Abstract Algebra (2nd Edition), Undergraduate Texts in Mathematics, Springer (SIE), Indian reprint, 2004

Theory of Automata			
Course Code:	**MAI 109**	**Credits:**	**3**
No. of Lectures (Hrs/Week):	**3**	**Mid Sem Exam Hours:**	**2**
Total No. of Lectures:	**45**	**End Sem Exam Hours:**	**3**

THEORY OF AUTOMATA

LEARNING OUTCOMES

- Acquire a fundamental understanding of the core concepts in automata theory and formal languages.
- An ability to design grammars and automata (recognizers) for different language classes.
- An ability to identify formal language classes and prove language membership properties.
- An ability to prove and disprove theorems establishing key properties of formal languages and automata.
- Acquire a fundamental understanding of core concepts relating to the theory of computation and computational models including decidability and intractability.

UNIT I INTRODUCTION
Introduction; alphabets, strings and languages; automata and grammars, deterministic finite automata (DFA)-formal definition, simplified notation: state transition graph, transition table, language of DFA, Nondeterministic finite Automata (NFA), NFA with epsilon transition, language of NFA, equivalence of NFA and DFA, minimization of finite automata, distinguishing one string from other, Myhill-Nerode Theorem

UNIT II REGULAR EXPRESSIONS
Regular expression (RE), definition, operators of regular expression and their precedence, algebraic laws for regular expressions, Kleen‟s theorem, regular expression to FA, DFA to regular expression, arden theorem, non regular languages, pumping lemma for regular languages. application of pumping lemma, closure properties of regular languages, decision properties of regular languages, FA with output: moore and mealy machine, equivalence of moore and mealy machine, applications and limitation of FA.

UNIT III CFG
Context Free Grammar (CFG) and Context Free Languages (CFL): definition, examples, derivation, derivation trees, ambiguity in grammar, inherent ambiguity, ambiguous to unambiguous CFG, useless symbols, simplification of CFGs, normal forms for CFGs: CNF and GNF, closure properties of CFLs, decision properties of CFLs: emptiness, finiteness and membership, pumping lemma for CFLs.

UNIT IV PUSH DOWN AUTOMATA
Push Down Automata (PDA): description and definition, instantaneous description, language of PDA, acceptance by final state, acceptance by empty stack, deterministic PDA, equivalence of PDA and CFG, CFG to PDA and PDA to CFG, two stack PDA

UNIT V TURING MACHINES (TM)
Basic model, definition and representation, instantaneous description, language acceptance by TM, variants of turing machine, TM as computer of integer functions, universal TM, church‟s thesis

recursive and recursively enumerable languages, halting problem, introduction to undecidability, undecidable problems about TMs. Post Correspondence Problem (PCP), modified PCP, introduction to recursive function theory.

References Books:

1. Hopcroft, Ullman, "Introduction to Automata Theory, Languages and Computation", Pearson Education
2. K.L.P. Mishra and N.Chandrasekaran, "Theory of Computer Science : Automata, Languages and Computation", PHI
3. Martin J. C., "Introduction to Languages and Theory of Computations", TMH
4. Papadimitrou, C. and Lewis, C.L., "Elements of the Theory of Computation", PHI

OPERATING SYSTEM

Operating System			
Course Code:	**MAI 111**	**Credits:**	**3**
No. of Lectures (Hrs/Week):	**3**	**Mid Sem Exam Hours:**	**2**
Total No. of Lectures:	**45**	**End Sem Exam Hours:**	**3**

Course Objectives:

- To introduce the different types of operating system.
- To introduce about the synchronization algorithms and semaphores.
- To introduce about conditional critical regions and continuous allocation.
- To introduce about the need for file system organization and disk scheduling.

Unit 1: Types of operating systems, Different views of the operating system, Principles of Design and Implementation. The process and threads, System programmer's view of processes, Operating system's views of processes, Operating system services for process management, Process scheduling, Schedulers, Scheduling algorithms, Overview of Linux operating system.

Unit 2: Interprocess synchronization, Mutual exclusion algorithms, Hardware support, Semaphores, Concurrent programming using semaphores.

Unit 3: Conditional critical regions, Monitors, Interprocess communication, Messages, Pipes. Deadlocks: Characterization. Prevention, Avoidance. detection and recovery, Combined approach to deadlock handling.

Unit 4: Contiguous allocation. Static and dynamic partitioned memory allocation, Segmentation, Non- contiguous allocation, Paging, Hardware support, Virtual Memory.

Unit 5: Need for files, File abstraction, File naming, File system organization, File system optimization, Reliability, Security and protection, I/O management and disk scheduling, Recent trends and developments.

Course Outcomes: On successful completion of this course, the students should be able to:

- Understand the different types of Operating systems and scheduling algorithms.
- Understand the synchronization algorithms and semaphores. • Appreciate the inter process communication and deadlock handling.
- Critically evaluate the different memory allocation techniques.
- Appreciate the importance of file system organization, I/O management and disk scheduling.

Text Books

1. Gary: Operating Systems- A modern Perspective, (2/e), Addison Wesley,2000.

2. M.Milenkovic: Operating systems, Concepts and Design, McGraw Hill,1992. Reference Books 1. C. Crowley: Operating Systems, Irwin,1997.

2. J.l. Peterson & A.S. Chatz: Operating System Concepts, Addison Wesley,1985.

3. W. Stallings: Operating Systems, (2/e), Prentice Hall,1995.

4. Mattuck,A., Introduction to Analysis,Prentice-Hall,1998. 5. Recent literature in Operating Systems.

DATA BASE MANAGEMENT SYSTEM

Data Base Management System			
Course Code:	**MDS 113/ MAI 113**	**Credits:**	**3**
No. of Lectures (Hrs/Week):	**3**	**Mid Sem Exam Hours:**	**2**
Total No. of Lectures:	**45**	**End Sem Exam Hours:**	**3**

Learning Outcomes:

- Understand database concepts and structures and query language
- Understand the E R model and relational model.
- To design and build a simple database system and demonstrate competence with the fundamental tasks involved with modeling, designing, and implementing a DBMS.
- Understand Functional Dependency and Functional Decomposition.
- Apply various Normalization techniques.
- Execute various SQL queries. Understand query processing and techniques involved in query optimization.
- Understand the principles of storage structure and recovery management.

UNIT I DATA BASE SYSTEM

Data base system vs. file system, view of data, data abstraction, instances and schemas, data models, ER model, relational model, database languages, DDL, DML, database access for applications programs, data base users and administrator, transaction management, data base system structure, storage manager, query processor, history of data base systems, data base design and ER diagrams, beyond ER design entities, attributes and entity sets, relationships and relationship sets, additional features of ER model, concept design with the ER model, and conceptual design for large enterprises.

UNIT II RELATIONAL DATA BASE MODEL

Introduction to the relational model, integrity constraint over relations, enforcing integrity constraints, querying relational data, and logical data base design, destroying /altering tables and views. relational algebra and calculus: relational algebra, selection and projection set operations, renaming, joins, division, relational calculus, tuple relational calculus, domain relational calculus, expressive power of algebra and calculus.

UNIT III SQL QUERY

Examples of basic SQL queries, nested queries, correlated nested queries set, comparison operators, aggregative operators, NULL values, comparison using null values, logical connectivity's, AND, OR and NOTR, impact on SQL constructs, outer joins, disallowing NULL values, complex integrity constraints in SQL triggers and active data bases.

UNIT IV NORMAL FORM

Problems caused by redundancy, decompositions, problem related to decomposition, reasoning about FDS, FIRST, SECOND, THIRD normal form, BCNF, forth normal form, fifth normal form, lossless join decomposition, dependency preserving decomposition, schema refinement in data base design, multi valued dependencies.

UNIT V TRANSACTION MANAGEMENT

ACID properties, transactions and schedules, concurrent execution of transaction, lock based concurrency control, performance locking, and transaction support in SQL, crash recovery, concurrency control, Serializability and recoverability, lock management, lock conversions, dealing with dead locks, specialized locking techniques, concurrency without locking, crash recovery: ARIES, log, other recovery related structures, the write, ahead log protocol, check pointing, recovering from a system crash, media recovery, other approaches and interaction with concurrency control.

References Books:

1. Elmasri Navrate, Data Base Management System, Pearson Education, 2008.
2. Raghurama Krishnan, Johannes Gehrke, Data Base Management Systems, TMH, 3rd edition, 2008.
3. C. J. Date, Introduction to Database Systems, Pearson Education, 2009.
4. Silberschatz, Korth, Database System Concepts, McGraw hill, 5th edition, 2005.
5. Rob, Coronel & Thomson, Database Systems Design: Implementation and Management, 2009.

C PROGRAMMING LAB

C Programming Lab			
Course Code:	**MDS 181**	**Credits:**	**3**
No. of Lectures (Hrs/Week):	**2**	**Mid Sem Exam Hours:**	**2**
Total No. of Lectures:	**30**	**End Sem Exam Hours:**	**3**

DATA BASE MANAGEMENT SYSTEM LAB

Data Base Management System Lab			
Course Code:	**MDS 113**	**Credits:**	**3**
No. of Lectures (Hrs/Week):	**3**	**Mid Sem Exam Hours:**	**2**
Total No. of Lectures:	**45**	**End Sem Exam Hours:**	**3**

Learning Outcomes:

- Understand database concepts and structures and query language
- Apply the basic concepts of Database Systems and Applications.
- Use the basics of SQL and construct queries using SQL in database creation and interaction.
- Design a commercial relational database system (Oracle, MySQL) by writing SQL using the system.
- Analyze and Select storage and recovery techniques of database system.

1 Write the queries for Data Manipulation and Data Definition Language.

2 Write SQL queries using logical operations and operators.

3 Write SQL query using group by function.

4 Write SQL queries for sub queries, nested queries

5 Write SQL queries to create views.

6 Write an SQL query to implement JOINS.

7 Write a query for extracting data from more than one table.

8. Write a query to understand the concepts for ROLL BACK, COMMIT & CHECK POINTS.

9. Create tables according to the following definition.

CREATE TABLE DEPOSIT (ACTNO VARCHAR2(5) ,CNAME VARCHAR2(18) , BNAME VARCHAR2(18) , AMOUNT NUMBER(8,2) ,ADATE DATE);
CREATE TABLE BRANCH(BNAME VARCHAR2(18),CITY VARCHAR2(18));
CREATE TABLE CUSTOMERS(CNAME VARCHAR2(19) ,CITY VARCHAR2(18));
CREATE TABLE BORROW(LOANNO VARCHAR2(5), CNAME VARCHAR2(18), BNAME VARCHAR2(18), AMOUNT NUMBER (8,2));

10. Retrieve all data from employee, jobs and deposit.
    (1) Give details of account no. and deposited rupees of customers having account opened between dates 01-01-06 and 25-07-06.
    (2) Display all jobs with minimum salary is greater than 4000.
    (3) Display name and salary of employee whose department no is 20. Give alias name to name of employee.
    (4) Display employee no,name and department details of those employee whose department lies in(10,20)

Semester II

JAVA PROGRAMMING			
Course Code:	**MAI102**	**Course Credits:**	**3**
Course Category:	**CC**	**Course (U / P)**	**P**
Course Year (U / P):	**1P**	**Course Semester (U / P):**	**2P**
No. of Lectures + Tutorials (Hrs./Week):	**03 + 00**	**Mid Sem. Exam Hours:**	**1**
Total No. of Lectures (L + T):	**45 + 00**	**End Sem. Exam Hours:**	**3**

COURSE OBJECTIVES

- To teach principles of object-oriented programming paradigm including abstraction, encapsulation, inheritance, and polymorphism.
- To impart fundamentals of object-oriented programming in Java, including defining classes, invoking methods, using class libraries, etc.
- To inculcate concepts of inheritance to create new classes from existing one & design the classes needed given a problem specification;
- To familiarize the concepts of packages and interfaces.
- To demonstrate the concept of event handling used in GUI.

COURSE OUTCOMES

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

- Demonstrate an introductory understanding of graphical user interfaces, multithreaded programming, and event-driven programming
- Analyze the necessity for Object Oriented Programming paradigm over structured programming and become familiar with the fundamental concepts in OOP like encapsulation, Inheritance and Polymorphism
- Design and develop java programs, analyze, and interpret object-oriented data and report results.
- Design an object-oriented system, AWT components and multithreaded processes as per needs and specifications.
- Prepare UML diagrams for software system

UNIT I OBJECT-ORIENTED PROGRAMMING

Concept of object-oriented programming (OOP), benefits of OOP, application of OOP, Java history, Java features, Java streaming, Java and Internet, Java contribution to Internet: Java applets, security, portability; Java environment, Java library, Java program structure, Java program, Java Virtual Machine (JVM) architecture, Just In Time compiler (JIT), data type, variables and arrays, operators, control statements, object-oriented paradigms; abstraction, encapsulation, inheritance, polymorphism, Java class and OOP implementation

UNIT II DATA TYPE, OPERATORS AND CONTROL STATEMENT

Data types, Java key words, identifiers, constants, variables, declaration and scope of the variable, symbolic constant, type casting, arithmetic operator, relational operator, logical operator, assignment operator, increment and decrement operator, conditional operator, bitwise operator, ?: operator, arithmetic expressions, expressions, type conversions in expressions, mathematical functions, more data types: arrays, strings, vectors, wrappers classes, program control statements: decision making and branching: if, if….else, else….if, else if ladder, switch, decision making and looping: while, do….while, for.

UNIT III CLASSES, OBJECTS AND METHODS

Java class libraries, class fundamentals, object, methods, adding variables, add methods, creating objects, accessing class members, constructors, methods overloading, static members, nesting of methods, inheritance: extending a class, overriding methods, final variables and methods, final classes, finalizer methods, abstract methods and classes, visibility control, exception handling fundamental.

UNIT IV INTERFACES AND PACKAGES

Interfaces, extending interfaces, implementing interfaces, interfaces references, accessing interface variable, creating queue interface, variable in interfaces, packages, finding a packages and classpath, package and member access, Java API package, system package, naming conventions, creating package, accessing a package, adding a class to a package, hiding classes,

UNIT V MULTITHREADING AND APPLET PROGRAMMING

Multithreading programming: creating threads, thread class and runnable interface extending the thread class, stopping and blocking a thread, life cycle of a thread, thread methods, thread exceptions, thread priority, synchronization, thread communication using notify(), wait(), and notify all(), applet programming : applet basic, applets architecture, a complete applet skeleton, building applets code, applets life cycle, creating a executable applet, designing a web page, applets tag, passing parameters to applets, applets and HTML.

Textbook:

1. Programming with JAVA, E. Balagurusawamy, Tata McGraw Hill, 1998.
2. JAVA Beginner's guide, Herbert Schildt, Tata McGraw Hill, 2007.
3. Java How to Program, Deitel & Deitel, Prentice-Hall, 1999.
4. The Complete Reference JAVA 2, Herbert Schildt, 5th Edition, Tata McGraw Hill, 2002.

5. The Complete Reference JAVA 2, Herbert Schildt, 7th Edition, Tata McGraw Hill, 2009.

Reference Books:

1. The Java Programming Language, Ken Arnold, James Gosling, Addison-Wesley, 1996.
2. How to Program Java, Peter Coffee, Ziff-Davis Press, 1996.

DATA STRUCTURE AND ALGORITHMS			
Course Code:	**MAI104**	**Course Credits:**	**3**
Course Category:	**CC**	**Course (U / P)**	**P**
Course Year (U / P):	**1P**	**Course Semester (U / P):**	**2P**
No. of Lectures + Tutorials (Hrs/Week):	**03 + 00**	**Mid Sem. Exam Hours:**	**1**
Total No. of Lectures (L + T):	**45 + 00**	**End Sem. Exam Hours:**	**3**
COURSE OBJECTIVES			
● To emphasize the importance of appropriate data structure in developing and implementing efficient algorithms			
● Understand basic data structures such as arrays, stacks, queues, hash tables and linked list			
● To analyze the asymptotic performance of various algorithms			
● Solve problems using graphs, trees and heaps			
● Apply important algorithmic design paradigms and methods of analysis			
COURSE OUTCOMES			
At the end of the course the students should be able to:			
● Define basic static and dynamic data structures and relevant standard algorithms for them.			

• Select basic data structures and algorithms for autonomous realization of simple programs or program parts.
• Determine and demonstrate bugs in program, recognise needed basic operations with data structures
• Formulate new solutions for programming problems or improve existing code using learned algorithms and data structures
• Evaluate algorithms and data structures in terms of time and memory complexity of basic operations.

UNIT I INTRODUCTION TO DATA STRUCTURES

Abstract data types, sequences as value definitions, data types in C, pointers in C, data structures and C, arrays in C, array as ADT, one dimensional array, Implementing one dimensional array, array as parameters, two dimensional array, structures in C, implementing structures, Unions in C, implementation of unions, structure parameters, allocation of storage and scope of variables, recursive definition and processes: factorial function, fibonacci sequence, recursion in C, efficiency of recursion, hashing: hash function, open hashing, closed hashing: linear probing, quadratic probing, double hashing, rehashing, extendible hashing.

UNIT II STACK, QUEUE AND LINKED LIST

Stack definition and examples, primitive operations, example -representing stacks in C, push and pop operation implementation, queue as ADT, C Implementation of queues, insert operation, priority queue, array implementation of priority queue, inserting and removing nodes from a list-linked implementation of stack, queue and priority queue, other list structures, circular lists: stack and queue as circular list - primitive operations on circular lists, header nodes, doubly linked lists, addition of long positive integers on circular and doubly linked list.

UNIT III TREES

Binary trees: operations on binary trees, applications of binary trees, binary tree representation, node representation of binary trees, implicit array representation of binary tree, binary tree traversal in C, threaded binary tree, representing list as binary tree, finding the Kth element, deleting an element, trees and their applications: C representation of trees, tree traversals, evaluating an expression tree, constructing a tree.

UNIT IV SORTING AND SEARCHING

General background of sorting: efficiency considerations, notations, efficiency of sorting, exchange sorts: bubble sort; quick sort; selection sort; binary tree sort; heap sort, heap as a priority queue, sorting using a heap, heap sort procedure, insertion sorts: simple insertion, shell sort, address calculation sort, merge sort, radix sort, sequential search: indexed sequential search, binary search, interpolation search.

UNIT V GRAPHS

Application of graph, C representation of graphs, transitive closure, Warshall's algorithm, shortest path algorithm, linked representation of graphs, Dijkstra's algorithm, graph traversal, traversal methods for graphs, spanning forests, undirected graph and their traversals, depth first traversal, application of depth first traversal, efficiency of depth first traversal, breadth first traversal, minimum spanning tree, Kruskal's algorithm, round robin algorithm.

Text Books:

1. Aaron M. Tenenbaum, Yeedidyah Langsam, Moshe J. Augenstein, 'Data structures using C', Pearson Education, 2004 / PHI.
2. E. Balagurusamy, 'Programming in Ansi C', Second Edition, TMH, 2003.
3. Robert L. Kruse, Bruce P. Leung Clovis L.Tondo, 'Data Structures and Program Design in C', Pearson Education, 2000 / PHI.

COMPILER DESIGN			
Course Code:	**MAI106**	**Course Credits:**	**2**
Course Category:	**CC**	**Course (U / P)**	**P**
Course Year (U / P):	**1P**	**Course Semester (U / P):**	**2P**
No. of Lectures + Tutorials (Hrs/Week):	**03 + 00**	**Mid Sem. Exam Hours:**	**1**
Total No. of Lectures (L + T):	**45 + 00**	**End Sem. Exam Hours:**	**3**
COURSE OBJECTIVES			
● Acquire knowledge of different phases and passes of the compiler and able to use the compiler tools like LEX, YACC, etc.			
● Understand the parser and its types i.e., Top-Down and Bottom-up parsers and construction of LL, SLR, CLR, and LALR parsing table.			
Understand backend of compiler: intermediate code, Code optimization Techniques and Error Recovery mechanisms			

● Understand backend of compiler: intermediate code, Code optimization Techniques and Error Recovery mechanisms
Understand issues of run time environments and scheduling for instruction level parallelism. ● Understand issues of run time environments and scheduling for instruction level parallelism.
COURSE OUTCOMES
At the end of the course the students should be able to:
● design different types of compiler tools to meet the requirements of the realistic constraints of compilers.
● Implement the compiler using syntax-directed translation method and get knowledge about the synthesized and inherited attributes.
● Acquire knowledge about run time data structure like symbol table organization and different techniques used in that.
● Understand the target machine's run time environment, its instruction set for code generation and techniques used for code optimization.
● . Understand and Implement a Parser.

UNIT I Introduction to Compiler

Phases and passes, Bootstrapping, Finite state machines and regular expressions and their applications to lexical analysis, Optimization of DFA-Based Pattern Matchers implementation of lexical analyzers, lexical-analyzer generator, LEX compiler, Formal grammars, and their application to syntax analysis, BNF notation, ambiguity, YACC. The syntactic specification of programming languages: Context free grammars, derivation, and parse trees, capabilities of CFG.

UNIT II Basic Parsing Techniques

Parsers, Shift reduce parsing, operator precedence parsing, top down parsing, predictive parsers Automatic Construction of efficient Parsers: LR parsers, the canonical Collection of LR(0) items, constructing SLR parsing tables, constructing Canonical LR parsing tables, Constructing LALR parsing tables, using ambiguous grammars, an automatic parser generator, implementation of LR parsing tables

UNIT III Syntax-directed Translation

Syntax-directed Translation schemes, Implementation of Syntax- directed Translators, Intermediate code, postfix notation, Parse trees & syntax trees, three address code,

quadruple & triples, translation of assignment statements, Boolean expressions, statements that alter the flow of control, postfix translation, translation with a top-down parser. More about translation: Array references in arithmetic expressions, procedures call, declarations, and case statements.

UNIT IV Symbol Tables

Data structure for symbols tables, representing scope information. Run-Time Administration: Implementation of simple stack allocation scheme, storage allocation in block structured language. Error Detection & Recovery: Lexical Phase errors, syntactic phase errors
semantic errors.

UNIT V Code Generation:

Design Issues, the Target Language. Addresses in the Target Code, Basic Blocks and Flow Graphs, Optimization of Basic Blocks, Code Generator. Code optimization: Machine-Independent Optimizations, Loop optimization, DAG representation of basic blocks, value numbers and algebraic laws, Global Data-Flow analysis.

Textbooks:

1. K. Muneeswaran,Compiler Design,First Edition,Oxford University Press.
2. P. Bennet, “Introduction to Compiler Techniques”, Second Edition, Tata McGraw-Hill,2003.
3. Henk Alblas and Albert Nymeyer, “Practice and Principles of Compiler Building with C”, PHI, 2001.
4. Aho, Sethi & Ullman, "Compilers: Principles, Techniques and Tools”, Pearson Education
5. V Raghvan, “ Principles of Compiler Design”, TMH
6. Kenneth Louden,” Compiler Construction”, Cengage Learning. Charles Fischer and Ricard LeBlanc,” Crafting a Compiler with C”, Pearson Education

EXPERT SYSTEM			
Course Code:	**MAI108**	**Course Credits:**	**3**
Course Category:	**CC**	**Course (U / P)**	**P**
Course Year (U / P):	**1P**	**Course Semester (U / P):**	**2P**
No. of Lectures + Tutorials (Hrs./Week):	**03 + 00**	**Mid Sem. Exam Hours:**	**1**
Total No. of Lectures (L + T):	**45 + 00**	**End Sem. Exam Hours:**	**3**

COURSE OBJECTIVES
● To study the idea of intelligent agents and search methods.
● To study about representing knowledge.
● To study the reasoning and decision making in uncertain world.
● To construct plans and methods for generating knowledge.
● To study the concepts of expert systems.
COURSE OUTCOMES
At the end of the course the students should be able to:
● Understands the basics of Artificial intelligence and the various searching techniques
● Analyses the knowledge representation through forward and backward chaining techniques
● Applies and articulate the knowledge over the design of semantic nets
● Evaluates the architecture and frameworks of Truth Maintenance systems through machine Learning algorithms
● Creates the design of real time AI and expert systems through various case studies

UNIT I

Define expert system, problem domain and knowledge domain, the advantages of an expert system, general stages in the development of an expert system, general characteristics of an expert system, history, and uses of expert systems today, rule-based expert systems, procedural and nonprocedural paradigms, characteristics of artificial neural systems. -The study of logic, difference between formal logic and informal logic, meaning of knowledge, how knowledge can be represented.

UNIT II

Semantic nets, how to translate semantic nets into PROLOG, limitations of semantic nets, schemas, frames, and their limitations, how to use logic and set symbols to represent knowledge, the meaning of propositional and first order predicate logic, quantifiers, imitations of propositional and predicate logic. Trees, lattices, and graphs, state, and problem spaces, AND-OR trees and goals, methods of inference, rules of inference, limitations of propositional logic, logic systems, resolution rule of inference, resolution systems, and deduction, shallow and causal reasoning, applying resolution to first-order predicate logic, forward and backward chaining, additional methods of reference, Meta knowledge, the Markov decision process.

UNIT III

Uncertainty and theories devised to deal with, types of errors attributed to uncertainty, errors associate, with induction, features of classical probability, experimental and subjective

probabilities, compound and conditional probabilities, hypothetical reasoning and backward induction, temporal reasoning.

UNIT IV

Markov chains, odds of belief, sufficiency and necessity, role of uncertainty in inference chains, implications of combining evidence, role of inference nets in expert systems, how probabilities are propagated. Sources of uncertainty in rules, methods of dealing with uncertainty, Dempster-Shafer theory, theory of uncertainty based on fuzzy logic, commercial applications of fuzzy logic. How to select an appropriate problem?

UNIT V

Stages in the development of an expert system, types of errors to expect in the development stages, the role of the knowledge engineer in the building of expert systems, the expected life cycle of an expert system, how to do a life cycle model.

Textbook:

[1] J. Giarratano and G. Riley, "Expert Systems -- Principles and Programming". 4th Edition, PWS Publishing Company, 2004.

[2] Durkin, J., Expert systems Design and Development, Macmillan, 1994 2. Elias M. Awad, Building Expert Systems, West Publishing Company 1996

COMPUTER VISION			
Course Code:	**MAI118**	**Course Credits:**	**2**
Course Category:	**CC-L2**	**Course (U / P)**	**P**

Course Year (U / P):	1P	Course Semester (U / P):	2P
No. of Lectures + Tutorials (Hrs/Week):	03 + 00	Mid Sem. Exam Hours:	1
Total No. of Lectures (L + T):	45 + 00	End Sem. Exam Hours:	3

COURSE OBJECTIVES
● To understand basics of graphics theorem and algorithms
● understand visual processing from both “bottom-up” (data oriented) and “top-down” (goals oriented) perspectives
● understand the roles of image transformations and their invariances in pattern recognition and classification
● understand in depth at least one important application domain, such as face recognition, detection, or interpretation
● understand basics of object detection and motion.

COURSE OUTCOMES
At the end of the course the students should be able to:
● decompose visual tasks into sequences of image analysis operations, representations, specific algorithms, and inference principles
● analyses the robustness, brittleness, generalizability, and performance of different approaches in computer vision
● to describe key aspects of how biological visual systems encode, analyze, and represent visual information
● be able to think of ways in which biological visual strategies might be implemented in machine vision, despite the enormous differences in hardware
● work in graphics industries with more knowledge.

UNIT I Digital Image Formation and low-level processing

Overview and State-of-the-art, Fundamentals of Image Formation, Transformation: Orthogonal, Euclidean, Affine, Projective, etc; Fourier Transform, Convolution and Filtering, Image Enhancement, Restoration, Histogram Processing.

UNIT II Depth estimation and multi-camera views

Perspective, Binocular Stereopsis: Camera and Epipolar Geometry; Homography, Rectification, DLT, RANSAC, 3-D reconstruction framework; Auto-calibration.

UNIT III Feature Extraction

Edges - Canny, LOG, DOG; Line detectors (Hough Transform), Corners - Harris and Hessian Affine, Orientation Histogram, SIFT, SURF, HOG, GLOH, Scale-Space Analysis- Image Pyramids and Gaussian derivative filters, Gabor Filters and DWT.

UNIT IV Image Segmentation

Region Growing, Edge Based approaches to segmentation, Graph-Cut, Mean-Shift, MRFs, Texture Segmentation, Object detection. Pattern Analysis Clustering: K-Means, K-Medoids,

Mixture of Gaussians, Classification: Discriminant Function, Supervised, Un-supervised, Semi-supervised; Classifiers: Bayes, KNN, ANN models; Dimensionality Reduction: PCA, LDA, ICA; Non-parametric methods.

UNIT V Motion Analysis

Background Subtraction and Modeling, Optical Flow, KLT, Spatio-Temporal Analysis, Dynamic Stereo; Motion parameter estimation. Shape from X: Light at Surfaces; Phong Model; Reflectance Map; Albedo estimation; Photometric Stereo; Use of Surface Smoothness Constraint; Shape from Texture, color, motion, and edges.

Text and Reference Books:

[1] Richard Szeliski, Computer Vision: Algorithms and Applications, Springer-Verlag London Limited 2011.
[2] Computer Vision: A Modern Approach, D. A. Forsyth, J. Ponce, Pearson Education, 2003.
[3] Richard Hartley and Andrew Zisserman, Multiple View Geometry in Computer Vision, Second Edition, Cambridge University Press, March 2004.
[4] Christopher M. Bishop; Pattern Recognition and Machine Learning, Springer, 2006
[5] R.C. Gonzalez and R.E. Woods, Digital Image Processing, Addison- Wesley, 1992.
[6] K. Fukunaga; Introduction to Statistical Pattern Recognition, Second Edition, Academic Press, Morgan Kaufmann, 1990.

DATA STRUCTURE AND ALGORITHMS LAB			
Course Code:	**MAI182**	**Course Credits:**	**2**
Course Category:	**CC**	**Course (U / P)**	**P**
Course Year (U / P):	**1P**	**Course Semester (U / P):**	**2P**

No. of Labs (Hrs/Week):	**03**	**Mid Sem. Exam Hours:**	
Total No. of Labs:	**10**	**End Sem. Exam Hours:**	**3**

COURSE OBJECTIVES
1. Introduce the concept of data structures through ADT including List, Stack, Queues .
2. To design and implement various data structure algorithms.
3. To introduce various techniques for representation of the data in the real world.
4. To develop application using data structure algorithms
5. Compute the complexity of various algorithms.
COURSE OUTCOMES
At the end of the course the students should be able to:
1. Select appropriate data structures as applied to specified problem definition
2. Implement operations like searching, insertion, and deletion, traversing mechanism etc. on various data structures.
3. Students will be able to implement Linear and Non-Linear data structures.
4. Implement appropriate sorting/searching technique for given problem.
5. Design advance data structure using Non-Linear data structure

List of Experiments:

1. Run time analysis of Fibonacci Series
2. Study and Application of various data Structure
3. Study and Implementation of Array Based Program
   a. Searching (Linear Search, Binary Search)
   b. Sorting (Bubble, Insertion, Selection, Quick, Merge etc)
   c. Merging
4. Implementation of Link List
   a. Creation of Singly link list, Doubly Linked list
   b. Concatenation of Link list
   c. Insertion and Deletion of node in link list
   d. Splitting the link list into two link list
5. Implementation of STACK and QUEUE with the help of
   a. Array
   b. Link List

6. Implementation of Binary Tree, Binary Search Tree, Height Balance Tree
7. Write a program to simulate various traversing Technique
8. Representation and Implementation of Graph
    a. Depth First Search
    b. Breadth First Search
    c. Prims Algorthim
    d. Kruskal's Algorithms
    e. Implementation of Hash Table

JAVA PROGRAMMING LAB			
Course Code:	**MAI184**	**Course Credits:**	**2**
Course Category:	**CC-L2**	**Course (U / P)**	**P**
Course Year (U / P):	**1P**	**Course Semester (U / P):**	**1P**
No. of Labs (Hrs/Week):	**01 (3 Hr)**		
Total No. of Labs :	**10**	**End Sem. Exam Hours:**	**3**

COURSE OBJECTIVES
● Knowledge of basic Object-Oriented paradigm, practices, and application.
● A general understanding of class, object, and methods.
● To make the student learn an object-oriented way of solving problem using java.
● To make the students, to write programs that connects to a database and be able to perform various operations.
● To make the students able to create the graphical user interface using Applets, AWT Components.

COURSE OUTCOMES
● At the end of the course the students should be able to:
● Basic knowledge and understanding of object-oriented programming.
● Ability to apply OOPs concept in real life problem.
● Ability to design, develop, maintain, and evaluate large-scale software systems.
● To produce efficient, reliable, robust, and cost-effective software solutions using Java.
● Ability to perform independent research and analysis.

1. Write a separate Java Code to implement each of the following:
   Class, Command Line Argument, how to enter value through keyboard

2. Write a separate Java code to implement each of the following data types: Variable, Constant, Arrays, Strings, Vectors, Wrapper Classes, Type Casting

3. Write a separate Java code to implement each of the following operators:
   Arithmetic operator, Relational operator, Logical operator, Assignment operator, increment & Decrement operator, Conditional operator, Bitwise operator, ? Operator

4. Write a separate Java code to implement each of the following control statements: Decision statement, Loop statement and Brach statements

5. Write a separate Java code to implement each of the following sorting: Bubble Sort, Selection sort, Insertion Sort, Merge sort

6. Write a separate Java Code to implement each of the following:
   Class, Object, Constructors, Method, Method overloading and Method overriding

7. Write a separate Java Code to implement each of the following: Final variable, final class, final method, abstract class, abstract method and concrete method

8. Write a separate Java code to implement each of the following: OOPs concepts: Abstraction, Polymorphism, Encapsulation, Inheritance

9. Write a separate Java code to implement each of the following: Exception handling with Try, Catch, Throws, Finally Multiple catch statement with the following exceptions: Arithmetic Exception, ArrayOutOfBoundsException and ArrayStoreException.

10: Write a separate Java code to implement each of the following: Interface and How to import Packages