

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Teaching and Research Methodology
Year: 2ndYear

Subject Code- MCSE301
Semester: Third

Module Number	Topics	Number of Lectures
1	Instruction	2L
	Introduction to content, Elements of instruction, Learning objectives,	1
	Roles of the teacher and the learner in instruction	1
2	Teaching and Learning	4L
	Application of theories of learning to teaching and learning, Sequence of learning and Strategies of learning,	2
	Teaching methods, their merits and demerits,	1
	Use of ICT in teaching & learning, Classroom management, Individual differences.	1
3.	Planning for teaching and learning	3L
	Understanding the syllabus, Preparation of a scheme of work,	2
	Lesson plan preparation, Micro teaching	1
4.	Assessment and Evaluation	4L
	Define measurement, assessment, test, evaluation, Purpose of assessment and evaluation,	2
	Types of tests, Grading and reporting the results assessment.	1
	Evaluating teaching and learning	1
5.	Definition and explanation of research	4L
	Types and Paradigms of Research, History and Philosophy of Research (esp. Philosophical evolution, pathways to major discoveries & inventions),	2
	Research Process decision, planning, conducting, Classification of Research Methods;	2
	Reflective Thinking, Scientific Thinking.	1
6.	Research problem formulation:	11L
	Literature review- need, objective, principles, sources, functions & its documentation,	2
	Problem formulation esp. sources, considerations & steps, Criteria of a good research problem, Defining and evaluating the research problem,	2
	Variables esp. types & conversion of concepts to variables. Research design esp. Causality, algorithmic, quantitative and qualitative designs,	2
	Various types of designs. Characteristics of a good	3

	research design, problems and issues in research design;	
	Hypotheses: Construction, testing, types, errors; Design of experiments especially classification of designs and types of errors.	2
7.	Problem solving:	5L
	Understanding the problem- unknowns, data & conditions, conditions - satisfiability, sufficiency, redundancy & contradiction,	1
	Separation of parts of the problem and conditions, notations; devising a plan- connection between data and unknown, similar/related problems, reuse of previous solutions, rephrasing/transforming the problem, solving partial or related problem,	2
	Transforming data and unknowns; carrying out the plan- esp. correctness of each step in multiple ways;	1
	Evaluation of solution and method- checking correctness of solution, different derivations, utility of the solution	1
8.	Data & Reports:	5L
	Infrastructural setups for research; Methods of data collection esp. validity and reliability, Sampling; Data processing and Visualization especially Classification;	2
	Ethical issues especially. bias, Misuse of statistical methods, Common fallacies in reasoning. Research Funding & Intellectual Property;	1
	Research reports: Research Proposal & Report writing esp. Study objectives, study design, problems and limitations;	1
	Prototype micro- project report implementing a major part of all the above (compulsory assignment)	1
Total Number Of Hours = 38		

Faculty In-Charge

HOD, CSE Dept.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Artificial Intelligence
Year: 4th Year

Subject Code-MCSE302A
Semester: Third

Module Number	Topics	Number of Lectures
1	Introduction:	2L
	Overview of Artificial intelligence- Problems of AI, AI technique, Tic - Tac - Toe problem.	
2	Intelligent Agents	2L
	Agents & environment, nature of environment, structure of agents, goal based agents, utility based agents, learning agents.	
3	Problem Solving	2L
	Problems, Problem Space & search: Defining the problem as state space search, production system, problem characteristics, issues in the design of search programs.	
4	Search techniques	5L
	Solving problems by searching: problem solving agents, searching for solutions; uniform search strategies: breadth first search, depth first search, depth limited search, bidirectional search, comparing uniform search strategies.	
5	Heuristic search strategies	5L
	Greedy best-first search, A* search, memory bounded heuristic search: local search algorithms & optimization problems: Hill climbing search, simulated annealing search, local beam search, genetic algorithms; constraint satisfaction problems, local search for constraint satisfaction problems	
6	Adversarial search	3L
	Games, optimal decisions & strategies in games, the mini max search procedure, alpha-beta pruning, additional refinements, iterative deepening.	
7	Knowledge & reasoning	3L
	Knowledge representation issues, representation & mapping, approaches to knowledge representation, issues in knowledge representation.	

8	Using predicate logic	2L
	Representing simple fact in logic, representing instant & ISA relationship, computable functions & predicates, resolution, natural deduction.	
9	Representing knowledge using rules	3L
	Procedural verses declarative knowledge, logic programming, forward verses backward reasoning, matching, control knowledge.	
10	Probabilistic reasoning	4L
	Representing knowledge in an uncertain domain, the semantics of Bayesian networks, Dempster-Shafer theory, Fuzzy sets & fuzzy logics.	
11	Planning	2L
	Overview, components of a planning system, Goal stack planning, Hierarchical planning, other planning techniques.	
12	Natural Language processing	2L
	Introduction, Syntactic processing, semantic analysis, discourse & pragmatic processing.	
13	Learning	2L
	Forms of learning, inductive learning, learning decision trees, explanation based learning, learning using relevance information, neural net learning & genetic learning.	
14	Expert Systems	2L
	Representing and using domain knowledge, expert system shells, knowledge acquisition	
Total Number Of Hours = 39		

Faculty In-Charge

HOD, CSE Dept.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Artificial Intelligence

Year: 4th Year

Assignments:

Subject Code-MCSE302A

Semester: Third

Module-I: Introduction

1. What do you mean by Artificial intelligence?
2. Explain Tic - Tac - Toe problem.

Module-II: Intelligent Agents

1. Explain nature of environment
2. Discuss the followings:
 - structure of agents
 - goal based agents
 - utility based agents
 - Learning agents

Module-III: Problem Solving

1. Explain how the problem as state space search has defined?
2. Define problem characteristics and issues in the design of search programs.

Module-IV: Search techniques

1. What do you mean by problem solving agents? searching for solutions
2. Explain depth limited search, bidirectional search.

Module-V: Heuristic search strategies

1. Explain Greedy best-first search
2. How Hill climbing search and simulated annealing search are different from each other?

Module-VI: Adversarial search

1. What do you mean by optimal decisions & strategies in games?
2. Explain the mini max search procedure, alpha-beta pruning.

Module-VII: Knowledge & reasoning

1. Explain different knowledge representation issues, representation & mapping.
2. Mention different approaches to knowledge representation. What are the issues in knowledge representation?

Module-VIII: Using predicate logic

1. How you represent simple facts in logic?

2. Explain ISA relationship, computable functions & predicates.

Module-IX: Representing knowledge using rules

1. Differentiate Procedural and declarative knowledge
2. Explain logic programming. What are the differences between forward and backward reasoning?

Module-X: Probabilistic reasoning

1. How you represent knowledge in an uncertain domain?
2. Explain the semantics of Bayesian networks. What do you mean by Dempster-Shafer theory?

Module-XI: Planning

1. Explain the components of a planning system. What is Goal stack planning?
2. What do you mean by Hierarchical planning?

Module-XII: Natural Language processing

1. Explain Syntactic processing in NLP.
2. What do you mean by semantic analysis?

Module-XIII: Learning

1. Explain the different forms of learning. What do you mean by inductive learning, learning decision trees, explanation based learning?
2. Differentiate neural net learning & genetic learning.

Module-XIV: Expert Systems

1. How do you representing and use domain knowledge?
2. Explain expert system shells, knowledge acquisition.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Bioinformatics
Year: 2nd Year

Subject Code: MCSE302B

Semester: 3rd

Module Number	Topics	Number of Lectures
1	Introduction to Molecular biology	5L
	1. Concepts of Cell, tissue, types of cell, components of cell, organelle. Functions of different organelles.	1
	2. Concepts of DNA: Basic Structure of DNA; Double Helix structure; Watson and crick model. Exons and Introns and Gene Concept. Concepts of RNA: Basic structure, Difference between RNA and DNA. Types of RNA	2
	3. Concept of Protein: Basic components and structure. Introduction to Central Dogma: Transcription and Tranlation, Introduction to Metabolic Pathways.	2
2	Sequence Databases	2L
	1. Introduction to Bioinformatics. Recent challenges in Bioinformatics. Protein Sequence Databases, DNA sequence databases.	1
	2. Sequence database search programs like BLAST and FASTA. NCBI different modules: GenBank; OMIM, Taxonomy browser, PubMed	1
3	DNA sequence analysis	14L
	1. DNA Mapping and Assembly : Size of Human DNA ,Copying DNA: Polymerase Chain Reaction (PCR)	2
	2. Hybridization and Microarrays, Cutting DNA into Fragments, Sequencing Short DNA Molecules	2
	3. Mapping Long DNA Molecules. DeBruijn Graph	3
	4. local and global alignment, pair wise and multiple alignment	2
	5. Dynamic Programming Concept	3
	6. Alignment algorithms: Needleman and Wunsch algorithm, Smith-Waterman.	
4	Introduction Probabilistic models used in Computational Biology	8L
	1. Probabilistic Models; Hidden Markov Model : Concepts, Architecture	2
	2. Transition matrix, estimation matrix. Application of HMM in Bioinformatics	2
	3. Gene finding, profile searches, multiple sequence alignment and regulatory site identification	2
	4. Bayesian networks Model :Architecture,	2

	Principle ,Application in Bioinformatics	
5	Biological Data Classification and Clustering	6L
	1. Assigning protein function and predicting splice sites	2
	2. Decision Tree Gene Expression Clustering	2
	3. K Means Algorithm.	2
Total Number Of Hours = 37		

Faculty In-Charge

HOD, CSE Dept.

Assignment:

Module-I: Introduction to Molecular biology

1. What is RNA? Explain its Basic structure. What are the difference between RNA and DNA?
2. Mention the basic components and structure of Protein.

Module-II: Sequence Databases

1. What is DNA sequence databases?
2. What do you mean by Bioinformatics? What are the latest challenges in Bioinformatics? Explain Protein Sequence Databases.

Module-III: DNA sequence analysis

1. Explain Polymerase Chain Reaction (PCR) and Sequencing Short DNA Molecules
2. What is DeBruijn Graph? Explain Smith-Waterman algorithm.

Module-IV: Introduction Probabilistic models used in Computational Biology

1. Explain the architecture of Hidden Markov Model. What are the application of HMM in Bioinformatics?
2. Explain the architecture of Bayesian networks Model.

Module-V: Biological Data Classification and Clustering

1. Describe Decision Tree Gene Expression Clustering.
2. Explain K Means Algorithm. Why we have to use this algorithm?

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Compiler Design
Year: 2nd Year MTech

Subject Code-MCSE302D
Semester: Third

Module Number	Course Details	Number of Lectures
UNIT 1 1	Introduction to Compiler:	4LH
	<ul style="list-style-type: none">• Compiler Construction tools• Analysis of the source program• The Phases of a Compiler• Cousins of the Compiler• Grouping of phases – Front and back ends, passes• Introduction, Types of translators	
2	Lexical Analysis:	6LH
	<ul style="list-style-type: none">• Role of Lexical Analyzer• Token, Patterns and Lexemes• Input buffering – buffer pairs and sentinels	
UNIT 2 3	Syntax Analysis:	7LH
	<ul style="list-style-type: none">• The role of a parser• Context free grammars, Writing a grammar• Top down Parsing• Non-recursive Predictive parsing (LL), Bottom up parsing, Handles• Viable prefixes• Operator precedence parsing• LR parsers (SLR, LALR), Parser generators (YACC)• Error Recovery strategies for different parsing techniques.	
4	Syntax directed translation:	7LH
	<ul style="list-style-type: none">• Syntax directed definitions• Construction of syntax trees• Bottom-up evaluation of S attributed definitions• L attributed definitions• Bottom-up evaluation of inherited attributes.	
UNIT 3 5	Type checking:	7LH
	<ul style="list-style-type: none">• Type systems• Specification of a simple type checker• Equivalence of type expressions• Type conversions	
	Run time environments:	

6	<ul style="list-style-type: none"> • Source language issues (Activation trees, Control stack, scope of declaration, Binding of names) • Storage organization (Subdivision of run-time memory, Activation records) • Storage allocation strategies • Parameter passing (call by value, call by reference, copy restore, call by name) • Symbol tables • Dynamic storage allocation techniques. 	5LH
UNIT 4	Intermediate code generation:	8LH
7	<ul style="list-style-type: none"> • Intermediate languages • Graphical representation • Three-address code • Implementation of three address statements (Quadruples, Triples, Indirect triples). 	
8	Code optimization and Code generations:	
	<ul style="list-style-type: none"> • Introduction • Basic blocks & flow graphs • Transformation of basic blocks • Dag representation of basic blocks, • The principle sources of optimization • Loops in flow graph • Peephole optimization • Code generations • Issues in the design of code generator, a simple code generator • Register allocation & assignment. 	
	Total Number Of Hours = 44	

Faculty In-Charge

HOD, CSE Dept.

Assignment:

Module-1(Introduction):

1. Find all strings in the language $(a+b)^*b(a+ab)^*$ of length less than 4.
2. With the help of a block diagram, show each phase of compiler including symbol table and error handling of a compiler.
3. Give the NFA for the following Regular Expression. Then find a DFA for the same language.
 $(a|b)^*abb$

Module-3 (Syntax Analysis):

1. Construct the Predictive Parsing table for the following grammar:

$$E \rightarrow E+ T|T$$

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Compiler Design

Subject Code-MCSE302D

Year: 2nd Year MTech

Semester: Third

T -> T* F|F

F-> (E) |id

2. Parse the following string by operator precedence parsing:

Id1+id2*id3

3. What are the main contributions of syntax directed translation in compiler? Design a dependency graph and direct acyclic graph for the string

a+a*(b-c)+(b-c)*d

4. What is operator precedence parsing? Discuss about the advantage and disadvantage of operator precedence parsing. consider the following grammar:

E-> TA

A-> +TA|€

T-> FB

B-> *FB|€

F-> id

Test whether this grammar is operator precedence grammar or not and show how the string w=id+id*id+id will be processed by this grammar.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Data Mining & Data Warehousing
Year: 2ndYear

Subject Code- MCSE302C
Semester: Third

Module Number	Topics	Number of Lectures
1	Overview of Data warehousing	4L
	Strategic information and the need for Data warehousing, ,.	1
	Defining a Data warehouse, Evolution of Data warehousing	2
	Data warehousing and Business Intelligence	1
2	The Building Blocks of Data warehouse	5L
	Defining features – Subject-oriented data, Integrated data, Time-variant data.	2
	Nonvolatile data, Data granularity Data warehouses and Data marts Architectural Types – Centralized, Independent data marts, Federated, Hub-and-Spoke, Data mart bus Overview of components - Source Data, Data Staging,	2
	Data Storage, Information Delivery, Metadata, and Management and Control components.	1
3.	Business Requirements and Data warehouse	6L
	Dimensional nature of Business data and Dimensional Analysis, Dimension hierarchies and categories, Key Business Metrics (Facts),	3
	Requirement Gathering methods and Requirements Definition Document (contents) Business Requirements and Data Design – Structure for Business Dimensions and Key Measurements, Levels of detail Business Requirements and the Architecture plan Business Requirements and Data Storage Specifications Business Requirements and Information Delivery Strategy	3
4.	Architectural components	7L
	Concepts of Data warehouse architecture – Definition and architecture in the areas of Data acquisition, Data storage, and Information delivery Distinguishing characteristics – Different objectives and scope.	2
	Data content, Complex analysis for faster response, Flexible and Dynamic, Metadata-driven etc. Architectural Framework – supporting flow of data, and the Management and Control module Technical architecture – Data acquisition, Data storage, and Information delivery Overview of the components of Architectural.	3
	Metadata types by functional areas – Data acquisition, Data storage, and Information delivery Business Metadata – overview of content and examples Technical Metadata – overview of content and examples Metadata Requirements, Sources of Metadata, Metadata management – challenges, Metadata Repository, Metadata integration and standards	2

5.	Matching information to classes of users	12L
	Information from Data warehouse versus Operational systems, Users of information – their needs and how to provide information, Information delivery – queries, reports, analysis, and applications, Information delivery tools – Desktop environment, Methodology and criteria for tool selection, Information delivery framework, Business Activity Monitoring, Dashboards and Scorecards,.	3
	OLAP in Data warehouse Overall concept of Online Analytical Processing (OLAP), OLAP definitions and rules, OLAP characteristics Major features and functions of OLAP – General features, Dimensional analysis, Hypercubes, Drill Down and Roll Up, Slice and Dice, Rotation, Uses and Benefits,	3
	Data Warehouse and the web Web-enabled Data Warehouse – adapting data warehouse for the web, Web-based information delivery – Browser technology for data warehouse and Security issues, OLAP and Web – Enterprise OLAP, Web-OLAP approaches, OLAP Engine design	3
	Data Mining Overview of Data mining – Definition, Knowledge Discovery Process (Relationships, Patterns, Phases of the process), OLAP versus Data mining, Some aspects of Data mining – Association rules, Outlier analysis, Predictive analytics etc), Concepts of Data mining in a Data warehouse environment, Major Data Mining techniques – Cluster Detection, Decision Trees, Memory-based Reasoning, Link Analysis, Neural Networks, Genetic Algorithms etc, Data Mining Applications in industry – Benefits of Data mining, Discussion on applications in Customer Relationship Management (CRM), Retail, Telecommunication, Biotechnology, Banking and Finance etc.	3
Total Number Of Hours = 34		

Faculty In-Charge

HOD, CSE Dept.

Assignment:

1. Define data warehouse? What are the characteristics of data warehouse?
2. Discuss the three-tier architecture of data warehouse.
3. What are the differences between data warehouse and data mart? What is virtual warehouse?
4. Explain K_Mean and DBSCAN clustering technique with proper example.
5. Describe the working principal of PAM algorithm.
6. Discuss different type of data warehousing schema with suitable diagram.
7. Explain the advantages and disadvantages of the decision tree approach over the approach of data mining?

8. Let us consider the following set of transaction in a book shop.

We shall look at asset of only 6 transaction of purchases of books. In the first transaction, purchases are made of book on cloud computing (CC), Data bases (D), Theory of Computation (TC), Computer Graphics (CG), Artificial Neural Network (ANN).

A: {ANN,CC,D,TC,CG}

T: {t₁,t₂,t₃,t₄,t₅,t₆}

t₁:={ANN, CC, TC, CG}

t₂:={CC, D, CG}

t₃:={ANN, CC, TC, CG}

t₄:={ANN, CC, D, CG}

t₅:={ANN, CC, D, TC, CG}

t₆:={CC, D, TC}

- (a) Find out the support of ANN, CC, D, TC, CG Individually.
- (b) Find out the Frequent sets at min. 60% confidence.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Microelectronics & VLSI Design
Year: 2nd Year

Subject Code-MCSE302E
Semester: Third

Module Number	Topics	Number of Lectures
1	Introduction to VLSI Design	6L
	1. VLSI Design Concepts, Moor's Law, Scale of Integration (SSI, MSI, LSI, VLSI, ULSI – basic idea only), Types of VLSI Chips (Analog & Digital VLSI)	1L
	2. Design styles- ASIC, PLA, PAL etc.	1L
	3. FPGA, Gate Array based design etc.	1L
	4. Top down, Bottom up, Semi custom, Full custom etc.	1L
	5. Design principles (Digital VLSI-Concept of Regularity etc.)	1L
	6. Design Domains (Behavioral, Structural, physical-Y chart)	1L
2	MOS Structures	12L
	1. E-MOS & D-MOS	1L
	2. Charge inversion in E-MOS	1L
	3. Threshold voltage, Flat-Band voltage, Potential balance & Charge balance	1L
	4. Inversion, MOS capacitances	1L
	5. three-terminal MOS structure with Body-effect	1L
	6. four-terminal MOS transistor: Drain current	1L
	7. I-V characteristics, Current-voltage equations (simple derivation),	1L
	8. scaling in MOSFET: General scaling, Constant voltage scaling & Constant field scaling	1L
	9. Short channel effects	2L
	10. CMOS inverter, Simple Combinational Gates-NAND gate and NOR gate using CMOS	2L
3	Micro-electronic Processes for VLSI Fabrication	10L
	1. Silicon Semiconductor Technology- An Overview, Wafer processing	1L
	2. Oxidation, Epitaxial deposition, Ion-implantation, Diffusion	1L
	3. Cleaning, Etching	1L
	4. Photo-lithography– Positive & Negative photo-resist	1L
	5. Basic CMOS Technology – Steps in fabricating CMOS	1L

	6. Basic n-well CMOS process, p-well CMOS process, Twin tub process	1L
	7. Silicon on insulator (SoI)	1L
	8. Layout Design Rules	1L
	9. Stick diagram with examples	1L
	10. Continue: Stick diagram with examples	1L
	Hardware Description Language	12L
4	1. Introduction, HDL and software languages, simulation, synthesis, VHDL, capabilities	1L
	2. Basic terminologies, entity, architecture,	1L
	3. Dataflow, structural, behavioural, mixed	1L
	4. Configuration declaration, package declaration, package body	1L
	5. Basic language elements	2L
	6. Details of Behavioural	L
	7. Details of Dataflow	1L
	8. Details of structural	L
Total Number Of Hours = 37L		

Angshuman Khan
Faculty In-Charge

Sandip Das
HOD, ECE Dept.

Assignment:

Module-1(Introduction to VLSI Design):

1. State Moore's law.
2. Describe VLSI design cycle. Why is it called cycle?
3. Write short note: regularity, modularity, locality.
4. What is hierarchical decomposition?
5. Describe 'Y' chart.
6. Discuss: CPLD, ROM, PLA, PAL, top-down & bottom-up approach, semicustom & full custom design.

Module-2 (MOS structure)

1. Describe accumulation, depletion, inversion, and pinch-off conditions of NMOS.
2. Draw C-V characteristics of MOS capacitor.
3. Discuss threshold voltage.
4. Derive the drain current equation of nMOS.
5. Design XOR gate using CMOS.
6. Prove that size of PMOS is 2.5 times of NMOS.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Module-3 (Micro-electronic Processes for VLSI Fabrication):

1. State Lambda rule and Micron rule?
2. Draw the lay-out of NAND2 and NOR2.
3. Draw the stick diagram of CMOS.
4. What is SOI and twin tub process?
5. Describe n-well fabrication process.
6. Describe the fabrication process of CMOS.

Module-4(HDL):

1. State the difference between hardware and software language.
2. What is the difference between synthesis and simulation?
3. What is the difference between variable and signal?
4. What is '9 value logic'?
5. Design D latch and D flipflop in dataflow style.
6. Write a short note on derived datatypes.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Advanced Engineering Mathematics
Year: 1st Year

Subject Code-MCSE101
Semester: first

Module Number	Topics	Number of Lectures (27)
1	Numerical Analysis:	4L
	Introduction to Interpolation formulae: Stirling Bessel's Spline	2
	Solution of system of linear and non-linear simultaneous equations: SOR algorithm Newton's method	2
2	Stochastic process:	8L
	Probability: review, random variables, random processes, Random walk, brownian motion, markov process,	4
	queues: (M/M/1):(/FIFO), (M/M/1):(N/FIFO)	4
3	Advanced linear algebra:	9L
	Vector spaces, linear transformations, eigenvalues, Eigenvectors, some applications of eigenvalue problems	4
	symmetric, skew-symmetric and orthogonal matrices, similarity of matrices	2
basis of Eigenvectors, diagonalisation	3	
4	Advanced Graph Theory:	6L
	Connectivity, Matching, Hamiltonian Cycles, Coloring Problems	4
	Algorithms for searching an element in a data structure (DFS, BFS)	2

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: **Advanced Operating System**
Year: **1st Year**

Subject Code: **MCSE 102**
Semester: **First**

Module Number	Topics	Number of Lectures
1	Operating System Introduction, Structures - Simple Batch, Multi programmed, time-shared, Personal Computer, Parallel, Distributed Systems, Real-Time Systems, System components, Operating-System services, System Calls, Virtual Machines, System Design and Implementation.	4L
	Process and CPU Scheduling - Process concepts and scheduling, Operation on processes, Cooperating Processes, Threads, and Interposes Communication Scheduling Criteria, Scheduling Algorithm, Multiple -Processor Scheduling, Real-Time Scheduling	5L
2	Memory Management and Virtual Memory - Logical versus Physical Address Space, Swapping, Contiguous Allocation, Paging, Segmentation, Segmentation with Paging. Demand Paging, Performance of Demanding Paging, Page Replacement, Page Replacement Algorithm, Allocation of Frames, Thrashing	6L
	File System Interface and Implementation - Access methods, Directory Structure, Protection, File System Structure, Allocation methods, Free-space Management, Directory Management, Directory Implementation, Efficiency and Performance	6L
3	Deadlocks - System Model, Dead locks Characterization, Methods for Handling Dead locks Deadlock Prevention, Deadlock Avoidance, Deadlock Detection, and Recovery from Deadlock	4L
	Process Management and Synchronization - The Critical Section Problem, Synchronization Hardware, Semaphores, and Classical Problems of Synchronization, Critical Regions, Monitors	5L

4	Operating System Security Issues- Introduction to the topic of Security in Operating Systems, Principles of Information Security, Access Control Fundamentals, Generalized Security Architectures	5L
5	Introduction to Distributed systems: Goals of distributed system, hardware and software Concepts, design issues	2L
	Elementary introduction to the terminologies within Modern Oss: Parallel, Distributed, Embedded & Real Time, Mobile, Cloud and Other Operating System Models	3L
Total Number Of Lectures = 40		

Faculty In-Charge

HOD, CSE Dept.

Assignments:

Module-1:

1. Explain Thread with real life examples.
2. Discuss real time scheduling.

Module-2:

1. What are the differences between paging and segmentation?
2. When does a page fault occur? Explain various page replacement strategies.

Module-3:

1. Why does starvation occur? What is the solution of starvation?
2. Discuss all the deadlock avoidance techniques?
3. State the roles of binary and count semaphore.

Module-4:

1. Discuss the security issues of OS?

Module-5:

1. Define real time operating system and its applications in real life?
2. Differentiate hard real time system from soft real time system.
3. Define distributed OS? How DOS differs from general purpose OS?
4. Write down the short notes on the following topics
 - A. POSIX
 - B. Event Driven Scheduling
 - C. Sporadic Task
 - D. VX-Works

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Advance Computer Architecture
Year: 1st Year

Subject Code: MCSE103
Semester: 1st

Module Number	Topics	Number of Lectures
1	The evolution of modern Computer systems & Introduction to high performance Computing	12
	1. DEC PDP-11, IBM 360/370 family, CDC Cyber 6600, Intel X86 architecture	2
	2. Performance measurement parameters – MIPS, MFLOPS, SPEC ratings, CPI etc	2
	3. Overview, Flynn's classifications – SISD, SIMD, MISD, MIMD	1
	4. Examples from Vector & Array Processors	1
	5. Performance comparison of algorithms for Scalar, Vector and Array Processor	2
	6. Fundamentals of UMA, NUMA, NORMA architectures Performance measurement for parallel architectures – Flynn's measure, Feng's measure, Handler's measure	2
	7. Amadahl's law of limitation for parallel processing, Gustafson's law	2
2	Pipelined processor design	12
	1. Pipeline performance measurement parameters – speedup factor, efficiency, throughput of a linear pipeline, comparing performance of a N stage pipeline with a N processor architecture	2
	2. Pipeline design principles – Uniform sub computations, Identical computations, Independent computations	2
	3. Examples from design of Arithmetic pipelines – Floating point Adders, Multipliers, Dividers etc.	2
	4. Classifications of Uni function, Multifunction & Dynamic pipelines	2
	5. Scheduling in a pipelines with feedback	2
	6. Pipeline hazards and their solutions	2
3	Various architectures	10
	1. RISC architecture, characteristics of RISC instruction set & RISC pipeline, its comparisons with CISC, necessity of using optimizing compilers with RISC architecture	2
	2. Examples from POWER PC and SPARC architectures	2
	3. Super pipelining (MIPS architecture),	2

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Advance Computer Architecture
Year: 1st Year

Subject Code: MCSE103
Semester: 1st

	Superscalar architecture	
	4. Diversified pipelines and out of order execution, VLIW architecture	2
	5. Hardware multithreading (Coarse grained, fine grained & simultaneous multithreading)	2
	Memory hierarchy	6
4	1. Techniques for improving Cache memory performance parameters, (reduce cache miss rate, reduce hit time, reduce miss penalty)	3
	2. Main memory performance enhancement – interleaved memory, improvement of memory bandwidth	2
	3. Use of TLB for performance enhancement	1
Total Lecture Hours – 34 l.h.		

Faculty In-Charge

HOD, CSE Dept.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Advance Computer Architecture
Year: 1st Year

Subject Code: MCSE103
Semester: 1st

Assignments:-

Unit 1:-

1. Explain the difference between hardwired and control and micro programmed control.
2. Explain the importance of different addressing modes in computer architecture with suitable example.
3. What is an instruction format? Explain different types of instruction formats in detail.
4. Explain Flynn's classification of computers.
5. Formulate a hardware procedure for detecting an overflow by computing the sign of the sum with the signs of the augends and addend. The numbers are in signed 2's complement representation.
6. Explain vector processing. What is the difference between vector & array processing?
7. What is the difference between serial and parallel transfer? Explain with the required example.
8. A digital computer has a common bus system for 16 register of 32 bits each. The bus is constructed with multiplexers.
 - (i) How many selection inputs are there in each multiplexer?
 - (ii) What sizes of multiplexers are needed?
 - (iii) How many multiplexers are there in the bus?
9. What is the difference between a direct and indirect address instruction? How many references to memory are needed for each type of instruction to bring an operand into a processor register?
10. A computer has 16 register, an ALU with 32 operations and a shifter with eight operations all Connected to a common bus system.
 - (i) Formulate a control word for a micro operation.
 - (ii) Specify the number of bits in each field on the control word and give a general encoding scheme.
11. List down the problems with MIPS system.
12. Discuss the Amadahl's law with proper example. Discuss fraction and speed up with real time case study.
13. Discuss NUMA and UMA with their properties.
14. Design a (very) simple CPU for an instruction set that contains only the following four instructions: lw (load word), sw (store word), add, and jump (unconditional branch). Assume that the instruction formats are similar to the MIPS architecture. If you assume a different format, state the instruction formats. Show all the components, all the links, and all the control signals in the datapath. You must show only the minimal hardware required to implement these four instructions. For each instruction show the steps involved and the values of the control signals for a single cycle implementation.

Unit 2:-

1. Draw a space time diagram for a six-segment pipeline showing the time it takes to process eight Tasks.
2. A no pipeline system takes 50 ns to process a task. The same task can be processed in 6 segment pipeline with a clock cycle of 10 ns. Determine the speedup ratio of pipeline for 100 tasks. What is maximum speedup ratio?
3. Explain hazards to the instruction pipeline with their solution.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Advance Computer Architecture
Year: 1st Year

Subject Code: MCSE103
Semester: 1st

4. What do you mean by instruction cycle and interrupt cycle? Draw the flowchart for instruction Cycle.
5. Determine the number of clock cycles that it takes to process 200 task in a six segment pipeline.
6. Consider the following MIPS assembly code:
LD R1, 45(R2)
ADD R7, R1, R5
SUB R8, R1, R6
OR R9, R5, R1
BNEZ R7, target
ADD R10, R8, R5
XOR R2, R3, R4
 - a) Identify each type of data dependency; list the two instructions involved; identify which instruction is dependent; and, if there is one, name the storage location involved.
 - b) Use MIPS five-stage pipeline (fetch, decode, register, execute, write-back) and assume a register file that writes in the first half of the clock cycle and reads in the second half cycle. Which of the dependencies that you found in part (a) become hazards and which do not? Why?

Unit 3:-

1. a) Name two RISC and two CISC processors. What are the main characteristics of RISC processors?
b) Define (i) superscalar and (ii) super-pipeline concepts. Derive the equation for ideal speedup for a superscalar super-pipelined processor compared to a sequential processor. Assume N instructions, k-stage scalar base pipeline, superscalar degree of m, and super pipeline degree of n.
2. Draw a diagram showing how the instruction fetch and execution units of a superscalar processor are connected. Show the widths of the datapath (in words - not bits; your diagram should be relevant to a 32-bit or 64-bit processor). Which factor primarily determines performance: the instruction issue width (number of instructions issued per cycle) or the number of functional units?
3. List the capabilities of the instruction fetch/despatch unit needed to make an effective superscalar processor.
4. Why does a VLIW machine need a good optimizing compiler?
5. Where can you find a small dataflow machine in every high performance processor?
6. How the processor performance can be improved other than the enhancement in VLSI technology?
7. Why does a VLIW machine restrict the op-codes which may be placed in any 'slot' of its instructions?

Unit 4:-

1. Give an example of a situation where caches in a shared memory machine need to be kept coherent.
2. List the states that a cache line may be in for the most commonly implemented cache coherence protocol. Provide a short description of each state.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Advance Computer Architecture

Subject Code: MCSE103

Year: 1st Year

Semester: 1st

3. What is the advantage of having both **Exclusive** and **Shared** states for cache lines in a shared memory machine?
4. What is the name of the programming model most commonly used for distributed memory machines?
5. A three-level memory system having cache access time of 5 nsec and disk access time of 40 nsec, has a cache hit ratio of 0.96 and main memory hit ratio of 0.9. What should be the main memory access time to achieve an overall access time of 16 nsec ?
6. According to the following information, determine size of the subfields (in bits) in the address for Direct Mapping and Set Associative Mapping cache schemes :
We have 256 MB main memory and 1 MB cache memory
The address space of the processor is 256 MB
The block size is 128 bytes
There are 8 blocks in a cache set.
7. Suppose physical addresses are 32 bits wide. Suppose there is a cache containing 256K words of data (not including tag bits), and each cache block contains 4 words. For each of the following cache configurations,
 - a. direct mapped
 - b. 2-way set associative
 - c. 4-way set associative
 - d. fully associativespecify how the 32-bit address would be partitioned. For example, for a direct mapped cache, you would need to specify which bits are used to select the cache entry and which bits are used to compare against the tag stored in the cache entry.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Adv. Algorithm
Year: 3rd Year

Subject Code-MCSE104
Semester: Fifth

Module Number	Topics	Number of Lectures
1	Complexity Analysis:	3L
	1. Time and space complexity. Asymptotic notations. Recurrence for divide and conquer And its solution, the substitution method and recursion-tree method for solving recurrences. The master method with proof	1
	2. Solve recursive function with different methods	2
2	Advanced data structure: :	3L
	1. ADT and data structure, linear vs non-linear data Structure. Tree: tree as an ADT, definition and terminologies, threaded binary tree, BST.	1
	2. Avl tree, balance multi way search tree: 2-3 tree, red- black tree, B tree, B+ tree, tries,spatial data representation using k-d tree, quad tree.	3
3	Algorithm Design Techniques:	15L
	1. Divide and Conquer: Basic method, use, Examples – Binary Search, Merge Sort ,Quick Sort and there complexity.	2
	2. Priority Queue:Definition, Heap Sort and its complexity.	1
	3. Dynamic Programming:Basic method, use, Examples – Matrix Chain Manipulation it's complexity,All pair shortest paths(Floyd-Warshall algorithm), single source shortest path(Bellman-ford algorithm), Travelling Salesman Problemand their complexities	4
	4. Backtracking:Basic method, use, Examples – 8 queens problem,Graph colouring problem.	1
	5. Branch and Bound:15 puzzle problem and its applications.	1
	6. Greedy Method:Basic method, use, Examples – Knapsack problem, Job sequencing with deadlines, Activity selection problem,single source shortest path (Dijkstra algorithm), Minimum cost spanning tree by Prim's and Kruskal's algorithm, their complexities	3
	7. Graph traversal algorithm: Breadth First Search (BFS) and Depth First Search	1

	(DFS) – complexity and comparison with different edges	
	8. Disjoint set manipulation:Set manipulation algorithm like UNION-FIND, union by rank, path compression	1
	9. Graph traversal algorithm:Breadth First Search (BFS) and Depth First Search (DFS) – complexity and comparison with different edges	1
4	Computational geometry:	8L
	1. Robust geometric primitives, convex hull, triangulation, voronoi diagrams, nearest neighbor search, range search, point location, intersection detection, bin packing,	4
	2. Medial-axis transform, polygon partitioning, simplifying polygons, shape similarity, motion planning, maintaining line arrangements, minkowski sum.	4
	Set and string problems:	5L
5	1. set cover, set packing, string matching, approximate string matching, text compression, cryptography, finite state machine minimization	3
	2. longestCommon substring/subsequence, shortest common superstring.	2
	Amortized Analysis:	1L
6	1. Aggregate, Accounting, and Potential Method.	1
	Advanced areas:	10L
7	1. P class, NP class, NP hard class, NP complete class	1
	2. Their interrelationship, Satisfiability problem, Cook’s theorem (Statement only), Clique decision problem, vertex cover problem, Hamiltonian cycle problem	2
	3. Necessity of approximation scheme, performance guarantee, polynomial time approximation schemes,	2
	4. Vertex Cover problem, travelling salesman problem	1
	5. Randomized algorithms, multithreaded algorithms, parallel algorithms.	4
Total Number Of Hours = 45		

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Assignment:

Module-1(Complexity analysis):

1. State the master theorem and prove it.
2. Solve the recurrence relation:
 - a) $T(n)=4T(n-1)+n^{1/2}$
 - b) $T(n)=2T(n^{1/2})+n\log n$

Module-2 (Advanced data structure):

1. Describe the insertion and deletion operation of red black tree and why it is required.
2. Define k-d tree and quad tree

Module-3(Algorithm Design Techniques):

1. Write down the algorithm of ternary search and find out the time complexity.
2. Modify the Merge sort algorithm so that the input array A is divided into K parts instead of 2. Analyze your algorithm. Assume $K > 1$.
3. Write an algorithm to sort an element in ascending order using min-heap and analyse it.
4. Find out the time complexity of heapify function in heap sort.
5. Explain Floyd warshall algorithm with example.
6. Consider the following five matrices:
 $A_1=2 \times 3, A_2=3 \times 4, A_3=4 \times 6, A_4=6 \times 2, A_5=2 \times 7$.
 - (i) How many parenthesization are possible to multiply these matrices?
 - (ii) Give a parenthesized expression for the order in which this optimal number of multiplications is achieved.
 - (iii) Find the optimal cost of the solution
7. Write down the complexity of N queens problem.
8. Draw the state space tree of 4 coloring problem.
9. Find out the time complexity of prime algorithm and dijkstra algorithm.
10. What are the features are present in any greedy algorithm?
11. Define path compression technique in union and find algorithm
12. How prim's algorithm follow this method explain with example
13. Find out the time complexity of BFS.
14. Write an algorithm to find the graph contain any back edge or not

Module-12(Computational geometry):

1. The convex hull of a set S is defined to be the intersection of all convex sets that contain S. For the convex hull of a set of points it was indicated that the convex hull is the convex set with smallest perimeter. We want to show that these are equivalent definitions. a. Prove that the intersection of two convex sets is again convex. This implies that the intersection of a finite family of convex sets is convex as well. b. Prove that the smallest perimeter polygon P containing a set of points P is convex. c. Prove that any convex set containing the set of points P contains the smallest perimeter polygon P.
2. Verify that the algorithm CONVEXHULL with the indicated modifications correctly computes the convex hull, also of degenerate sets of points. Consider for example such nasty cases as a set of points that all lie on one (vertical) line.

Module-12 (Set and string problems):

1. Explain KMP algorithm with example. Why KMP is better than Naïve and string matching with finite automata.
2. Write an algorithm to find out the longest common sub sequence in a string.

Module-13(Amortize analysis):

1. Define Aggregate Method Accounting Method and Potential Method.

Module-15(Advanced areas):

1. Show all satisfiability problem are verifiable in polynomial time.
2. Prove that approx-vertex-cover is 2-approximation algorithm
3. Write a short note on Randomized algorithms, multithreaded algorithms, parallel algorithms.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Artificial Neural Network
Year: 1st Year

Subject Code- MCS105A
Semester: First

Module Number	Topics	Number of Lectures
1	Introduction to artificial neural networks :	5L
	1. Biological neural networks, Pattern analysis tasks: Classification, Regression, Clustering.	2
	2. Computational models of neurons.	1
	3. Structures of neural networks	1
	4. Learning principles	1
2	Linear models for regression and classification:	8L
	1. Polynomial curve fitting, Bayesian curve fitting.	2
	2. Linear basis function models, Bias-variance decomposition.	2
	3. Bayesian linear regression, Least squares for classification.	2
	4. Logistic regression for classification, Bayesian logistic regression for classification.	2
3.	Feed forward neural networks :	8L
	1. Pattern classification using perceptron, Multilayer feed forward neural networks (MLFFNNs).	2
	2. Pattern classification and regression using MLFFNNs.	2
	3. Error back propagation learning.	1
	4. Fast learning methods: Conjugate gradient method, Auto associative neural networks.	2
	5. Bayesian neural networks.	1
4	Radial basis function networks :	5L
	1. Regularization theory.	2
	2. RBF networks for function approximation.	2
	3. RBF networks for pattern classification.	1
5	Self-organizing maps :	4L
	1. Pattern clustering, Topological mapping	2
	2. Kohonen's self-organizing map.	2
6	Feedback neural networks :	5L
	1. Pattern storage and retrieval, Hopfield model	2
	2. Boltzmann machine	1

	3. Recurrent neural networks.	2
7	Kernel methods for pattern analysis :	8L
	1. Statistical learning theory	2
	2. Support vector machines for pattern classification	2
	3. Support vector regression for function approximation	2
	4. Relevance vector machines for classification and regression	2
Total Number Of Hours = 43		

Faculty In-Charge

HOD, CSE Dept.

Assignment:

Module-1(Introduction):

1. Write Down short Notes: Classification & Regression.
2. Draw the structure of Neural networks & describe it.

Module-2(Linear models for regression and classification):

1. Notes: Bayesian linear regression & Bias-variance decomposition.
2. Prove that: $MSE = Bias^2 + Var$.

Module-3(Feed forward neural networks):

1. Short Notes: Bayesian neural networks.
2. Describe the Error back propagation learning.
3. Write algorithm of back-propagation rule

Module-4(Radial basis function networks):

1. Write down short notes: RBF networks.

Module-5(Self-organizing maps):

1. Consider a Kohonen net with two cluster (outputs) units & five input units. The weight vectors for the output units are $W1=[1,0.8,0.6,0.4,0.2]$ and $W2=[1,0.5,1,0.5,1]$. Use the square of the Euclidean distance to find the winning neuron for the input pattern $X=[0.5,1,0.5,0,0.5]$. Find the new weights for the winning unit. Assume learning rate as 0.2.

Module-6(Feedback neural networks):

1. Short Notes: Boltzmann machine & Recurrent neural networks.

Module-7(Kernel methods for pattern analysis):

1. Write down short note for SVM.
2. How its works in classification & regression?

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Agent Based Intelligent Systems

Subject Code-MCSE105B

Year: 1st Year

Semester: First

Module Number	Topics	Number of Lectures
1	Introduction:	9L
	1. Definitions, Foundations, History, Intelligent Agents	3L
	2. Problem Solving, Searching, Heuristics	3L
	3. Constraint Satisfaction Problems, Game playing.	3L
2	Knowledge Representation And Resoning:	9L
	1. Logical Agents, First order logic, First Order Inference	3L
	2. Unification, Chaining, Resolution	2L
	3. Strategies, Knowledge Representation	3L
	4. Objects, Actions, Events.	1L
3	Planning Agent:	9L
	1. Planning Problem, State Space Search	2L
	2. Partial Order Planning, Graphs, Nondeterministic Domains	3L
	3. Conditional Planning, Continuous Planning, Multi-Agent Planning.	4L
4	Agent And Uncertainty:	9L
	1. Acting under uncertainty – Probability Notation, Bayes Rule and use	3L
	2. Bayesian Networks, Other Approaches	2L
	3. Time and Uncertainty, Temporal Models	2L
	4. Utility Theory, Decision Network – Complex Decisions.	2L
5	Higher Level Agent:	9L
	1. Knowledge in Learning, Relevance Information	2L
	2. Statistical Learning Methods, Reinforcement Learning	3L
	3. Communication, Formal Grammar, Augmented Grammars, Future of AI.	4L

Faculty In-Charge

HOD, CSE Dept.

Assignment:

Module-1(Introduction):

1. Intelligent Agents
2. Constraint Satisfaction Problems and Game playing.

Module-2 (Knowledge Representation And Resoning):

1. Logical Agents
2. First order logic and First Order Inference

Module-3(Planning Agent):

1. State Space Search
2. Different planning

Module-4(Agent And Uncertainty)

1. Probability Notation
2. Bayesian Networks

Module-5(Higher Level Agent):

1. Reinforcement Learning

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Advanced Soft Computing
Year: 1st Year

Subject Code-MCSE105C
Semester: First

Module Number	Topics	Number of Lectures
1	Introduction:	3L
	1. Introduction to Soft Computing; Difference between Hard and Soft Computing;	1
	2. Introduction to Fuzzy Systems, Artificial Neural Network, Evolutionary Algorithms, Rough Set Theory; Hybrid Systems	2
2	Fuzzy Sets & Logic	10L
	1. Introduction to Fuzzy Sets, Classical and Fuzzy Sets.	1
	2. Fuzzy Sets - Membership Function, Basic Operations, Linguistic Variable, Properties.	2
	3. Fuzzy relations - Cartesian product, Operations on relations; Crisp logic— Laws of propositional logic, Inference.	2
	4. Predicate logic—Interpretations, Inference, Fuzzy logic—Quantifiers, Inference.	2
	5. Fuzzy Rule based system; De-fuzzification methods; Basic Applications of Fuzzy Sets and Logics.	3
3	Pattern Recognition	3L
	1. Pattern Classification, Pattern Association, Clustering.	1
	2. Simple Clustering algorithm, k-means & k-medoid based algorithm.	2
4	Artificial Neural Network	13L
	1. Neural Networks: Introduction.	1
	2. Mathematical Models, ANN architecture.	1
	3. Learning rules, Supervised, Unsupervised and reinforcement Learning.	2
	4. Multilayer Perception, Applications of Artificial Neural Networks.	2
	5. Competitive learning networks, Kohonen self organizing networks, Hebbian learning.	3
6. Hopfield Networks, Associative Memories, The boltzman machine; Applications of ANN	4	
5	Genetic Algorithms	10L
	1. Introduction, Single and Multi-Objective Optimization, Encoding, Fitness Function, Genetic Operations, Genetic Parameters; Genetic Algorithm; Basic Applications.	5
	2. Schema theorem; Convergence Theory; Multiobjective optimization using GA	5

	(MOGA); Non-Dominated Sorting	
6	Hybrid Systems	6L
	1. Hybrid systems, GA based ANN (Optimal Weight Selection).	3
	2. Neuro Fuzzy Systems, fuzzy Neuron, architecture, learning, application.	3

Faculty In-Charge

HOD, CSE Dept.

Assignment:

Module-1(Introduction):

1. Difference between Hard and Soft Computing.
2. Artificial Neural Network
3. Evolutionary Algorithms
4. Rough Set Theory
5. Hybrid Systems

Module-2 (Fuzzy Sets & Logic):

1. Classical and Fuzzy Sets
2. Crisp logic—Laws of propositional logic.
3. Predicate logic
4. Basic Applications of Fuzzy Sets and Logics.

Module-3(Pattern Recognition):

1. k-means & k-medoid based algorithm.

Module-4(Artificial Neural Network)

1. Supervised, Unsupervised and reinforcement Learning
2. Multilayer Perception
3. Hopfield Networks

Module-5(Genetic Algorithms):

1. Genetic Algorithm.
2. Convergence Theory

Module-6(Hybrid Systems):

1. Fuzzy Neuron

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Object Oriented Information System Design

Subject Code-MCSE105D

Year: 1st Year Semester: 1st

Module Number	Topics	Number of Lectures
1	Data and Information	3L
	1. Types of information: operational, tactical, strategic and statutory –	1
	2. why do we need information systems – management structure – requirements of information at different levels of management – functional allocation of management	1
	3. requirements of information for various functions – qualities of information – small case study	1
2	Systems Analysis and Design Life Cycle	3L
	1. Requirements determination – requirements specifications	1
	2. feasibility analysis – final specifications – hardware and software study – system design – system implementation	1
	3. System evaluation – system modification. Role of systems analyst – attributes of a systems analyst – tools used in system analysis	1
3	Information	3L
	1. Information gathering	1
	2. strategies – methods – case study – documenting study – system requirements specification	1
	3. From narratives of requirements to classification of requirements as strategic, tactical, operational and statutory. Example case study	1
4	Feasibility analysis	6L
	1. Feasibility analysis – deciding project goals – examining alternative solutions – cost	2
	2. benefit analysis – quantifications of costs and benefits – payback period – system proposal preparation for managements	2
	3. parts and documentation of a proposal – tools for prototype creation	2
5	Tools for systems analysts	3L
	1. data flow diagrams –	1
	2. case study for use of DFD, good conventions – leveling of DFDs – leveling rules	1
	3. logical and physical DFDs – software tools to create DFDs	1

6	Structured systems analysis and design	4L
	<ol style="list-style-type: none"> 1. Structured systems analysis and design – procedure specifications in structured English – examples and cases 2. decision tables for complex logical specifications – specification oriented design vs procedure oriented design 	<p style="text-align: center;">2</p> <p style="text-align: center;">2</p>
7	Data oriented systems design	6L
	1. entity relationship model – E-R diagrams	2
	<ol style="list-style-type: none"> 2. relationships cardinality and participation – normalizing relations 3. various normal forms and their need – some examples of relational data base design 	2
8	Data input methods	3L
	1. coding techniques	1
	2. requirements of coding schemes – error detection of codes	1
	3. validating input data – input data controls interactive data input	1
Total Number Of Hours = 31		

Faculty In-Charge

HOD, CSE Dept.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Object Oriented Information System Design

Subject Code-MCSE105D

Year: 1st Year Semester: 1st

Assignment:

Module-I:

1. Why do we need information systems? Explain management structure.
2. Explain the requirements of information at different levels of management.

Module-II:

1. Explain Requirements determination and requirements specifications with proper example.
2. What do you mean by feasibility analysis? Explain with proper example.

Module-III:

1. Explain Information gathering strategies.
2. What do you mean by system requirements specification?

Module-IV:

1. Explain Feasibility analysis.
2. What is payback period?

Module-V:

1. What is DFD?
2. Explain logical and physical DFDs.

Module-VI:

1. What is structured systems analysis and design?
2. Differentiate specification oriented design and procedure oriented design system.

Module-VII:

1. What do you mean by E-R diagram? Explain with an example.
2. What is relationships cardinality?

Module-VIII:

1. Explain the requirements of coding schemes. Why error detection of codes is required?
2. How you validate the inputted data?

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Software Engineering & CASE tools
Year: 1st Year

Subject Code-MCSE105E
Semester: First

Module Number	Topics	Number of Lectures
1	Software Engineering	6L
	1. Objectives, Definitions	1
	2. Software Process models - Waterfall Model, Prototype model, RAD	3
	3. Evolutionary Models	1
	4. Incremental, Spiral.	1
2	Software Project Planning	4L
	1. Feasibility Analysis, Technical Feasibility	2
	2. Cost- Benefit Analysis,	1
	3. COCOMO model.	1
3.	Structured Analysis	5L
	1. Context diagram and DFD, Physical and Logical DFDs.	3
	2. Data Modelling, ER diagrams, Software Requirements Specification.	2
4	Design Aspects	4L
	1. Top-Down And Bottom-Up design; Decision tree, decision table and structured English	2
	2. Structure chart, Transform analysis Functional vs. Object- Oriented approach.	2
5	Unified Modelling Language	6L
	1. Class diagram, interaction diagram: state chart diagram,	2
	2. collaboration diagram, sequence diagram,	2
	3. Activity diagram, implementation diagram.	2
6	Coding & Documentation	6L
	1. Structured Programming, Modular Programming, Module Relationship-Coupling,	3
	2. Cohesion, OO Programming, Information Hiding, Reuse, System Documentation.	3
7	Testing	4L
	1. Levels of Testing	1
	2. Integration Testing	1
	3. System Testing.	1
	Software Quality	4L
	1. Quality Assurance	1

8	2. Software Maintenance	1
	3. Software Configuration Management	1
	4. Software Architecture, Computer Aided Software Engineering (CASE) tool	2
9	Object modeling and design	10
	Classes, objects, relationships, key abstractions, common mechanisms, diagrams, class diagrams, advanced classes, advanced relationships,	3
	interfaces, types, roles, packages, instances, object diagrams, interactions, use cases, use case diagrams, interaction diagrams, activity diagrams, events and signals, state machines,	3
	processes, threads, state chart diagrams, components, deployment, collaborations, patterns and frameworks, component diagrams, systems and models, code generation and reverse engineering	6
Total Number Of Hours = 49		

Faculty In-Charge

HOD, CSE Dept.

Assignment:

Module-1(Software Engineering):

1. Explain Software Engineering as a layered technology with a neat sketch.
2. What are process models why do we require them? Explain in detail any one of the process model?
3. Why it is important to design a software and explain the role it play?

Module-2 (Software Project Planning):

1. What are the purposes of Data Flow diagrams, Entity-Relationship diagrams? Give an example diagram of each. (10 mks)

Module-3(Structured Analysis):

1. What is functional and non-functional requirements?
2. What is a requirement modeling? Explain about the types of requirement modeling
3. How can software project estimation be done by empirical estimation model

Module-4(Design Aspects):

1. How designing is done by functional based component design?

Module-5(Unified Modelling Language):

1. Give a brief description about class hierarchies and class based component design?

Module-6(Coding & Documentation):

Module-7(Testing):

1. What is user acceptance testing? Explain different testings in user acceptance testing. Why is it necessary?

Module-8(Software Quality):

1. Explain about the architecture of software and its importance?

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Computer Graphics
Year: 3rd Year

Subject Code-CS604B
Semester: Sixth

Module Number	Course Details	Number of Lectures
UNIT 1	Introduction to computer graphics & graphics systems:	4LH
1	<ul style="list-style-type: none">• Overview of computer graphics, representing pictures, preparing, presenting & interacting with pictures for presentations• Visualization & image processing; RGB color model, direct coding, lookup table• storage tube graphics display, Raster scan display, 3D viewing devices, Plotters, printers, digitizers, Light pens etc.• Active & Passive graphics devices	
2	Scan conversion:	6LH
	<ul style="list-style-type: none">• Points & lines, Line drawing algorithms• DDA algorithm, Bresenham's line algorithm, Circle generation algorithm• Ellipse generating algorithm; scan line polygon, fill algorithm• boundary fill algorithm, flood fill algorithm.	
UNIT 2	2D transformation & viewing:	7LH
3	<ul style="list-style-type: none">• Basic transformations: translation, rotation, scaling;• Matrix representations & homogeneous coordinates, transformations between coordinate systems;• Reflection shear; Transformation of points, lines, parallel lines, intersecting lines.• Cohen and Sutherland line clipping, Sutherland-Hodgeman Polygon clipping, Cyrus-beck clipping method	
4	3D transformation & viewing:	7LH
	<ul style="list-style-type: none">• 3D transformations: translation, rotation, scaling & other transformations• Rotation about an arbitrary axis in space• Clipping, view port clipping, 3D viewing.	
UNIT 3	Curves and Hidden surfaces:	

5	<ul style="list-style-type: none"> • Curve representation, surfaces, designs, Bezier curves • B-spline curves, end conditions for periodic B-spline curves, rational B-spline curves. • Depth comparison, Z-buffer algorithm, Back face detection • BSP tree method, the Painter's algorithm, scan-line algorithm • Hidden line elimination, wire frame methods , fractal - geometry. 	7LH
6	<p>Introduction to Ray-tracing:</p> <ul style="list-style-type: none"> • Human vision and color • Lighting, Reflection and transmission models. 	8LH
UNIT 4 7	<p>Multimedia:</p> <ul style="list-style-type: none"> • Introduction to Multimedia: Concepts, uses of multimedia, hypertext and hypermedia • Image, video and audio standards. Audio: digital audio, MIDI, processing sound, sampling, compression • Video: MPEG compression standards, compression through spatial and temporal redundancy, inter-frame and intra-frame compression. Animation: types, techniques, key frame animation, utility, morphing • Virtual Reality concepts. 	7LH
	Total Number Of Hours = 44	

Faculty In-Charge

HOD, CSE Dept.

Assignment:

Module-1(Introduction to computer graphics & graphics systems):

1. A monochromatic graphic display system has 525 scan lines with an aspect ratio 9:16. If each pixel is displaceable in 512 shades
 - (i) How many pixels are displayed on the screen?
 - (ii) What is the picture storage memory size?
2. What do you mean by window and viewport? Describe the relationship for window to viewport mapping.

Module-3 (2D transformation & viewing):

1. Prove that successive scaling is multiplicative
2. Write down mid-point ellipse drawing algorithm

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Computer Graphics

Subject Code-CS604B

Year: 3rd Year

Semester: Sixth

3. a rectangular 2D clipping window has its lower left corner at (100,10) and upper right corner at (160,40). Find visible portion of lines A(50,0), B(120,30) and C(120,20), D(140,80) using mid point sub division algorithm.

Module-4 (Curves and Hidden surfaces)

1. Write down the procedure for drawing B-spline curves and also write down its property
2. Derive the condition to be satisfied when joining two Bezier curves with second order continuity at the joints.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Operating System Lab

Course Code: MCSE191

L-T-P scheme: 0-0-3

Course Credit: 2

Objectives:

1. To learn and understand system calls (remote procedure calls) related to files, processes, threads, signals, semaphores and implement system programs based on that.
2. To provide an understanding of the design aspects of operating system.
3. To provide an efficient understanding of the language translation peculiarities by designing a complete translator for a mini language.

Learning Outcomes: The students will have a detailed knowledge of the concepts of process and shared memory, aware of a variety of approaches to process management and main-memory management, including interference, deadlock, scheduling, fragmentation, thrashing, learn the basics behind file systems and input output systems and understand the fundamentals of network and distributed operating systems. Upon the completion of Operating Systems practical course, the student will be able to:

- **Understand** and implement basic services and functionalities of the operating system using system calls.
- **Use** modern operating system calls and synchronization libraries in software/ hardware interfaces.
- **Understand** the benefits of thread over process and implement synchronized programs using multithreading concepts.
- **Analyze** and simulate Deadlock Avoidance and Protection algorithm like Bankers.
- **Implement** memory management schemes
- **Implement** remote procedure call
- **Understand** producer and consumer problem.

Course Contents:

Exercises that must be done in this course are listed below:

Exercise No.1: Simulate Banker's Algorithm for Dead Lock Avoidance

Exercise No.2: Simulate Banker's Algorithm for Dead Lock Prevention

Exercise No. 3: Simulate Paging Technique of Memory Management

Exercise No. 4: Thread Creation

Exercise No. 5: Process Creation

Exercise No. 6: Producer and Consumer Problem

Exercise No. 7: Implementation of Remote Procedure Call

Text Book:

1. Maurice J. Bach, Design of the UNIX Operating System, PHI.

Recommended Systems/Software Requirements:

1. Intel based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
2. Turbo C or TC3 compiler in Windows XP or Linux Operating System.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment No: 1 BANKER'S DEADLOCK AVOIDANCE

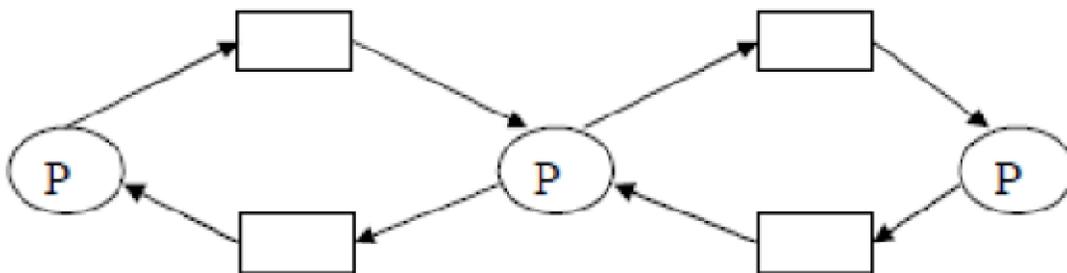
Aim: To Simulate Bankers Algorithm for Deadlock Avoidance.

Description:

Deadlock: A process request the resources, the resources are not available at that time, so the process enter into the waiting state. The requesting resources are held by another waiting process, both are in waiting state, this situation is said to be Deadlock. A deadlocked system must satisfied the following 4 conditions. These are:

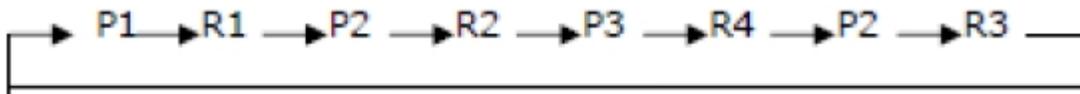
(i) Mutual Exclusion: Mutual Exclusion means resources are in non-sharable mode only, it means only one process at a time can use a process.

(ii) Hold and Wait: Each and every process is the deadlock state, must holding at least one resource and is waiting for additional resources, that are currently being held by another process.



(iii) No Preemption: No Preemption means resources are not released in the middle of the work, they released only after the process has completed its task.

(iv) Circular Wait: If process P1 is waiting for a resource R1, it is held by P2, process P2 is waiting for R2, R2 held by P3, P3 is waiting for R4, R4 is held by P2, P2 waiting for resource R3, it is held by P1.



Deadlock Avoidance: It is one of the method of dynamically escaping from the deadlocks. In this scheme, if a process request for resources, the avoidance algorithm checks before the allocation of resources about the state of system. If the state is safe, the system allocate the resources to the requesting process otherwise (unsafe) do not allocate the resources. So taking care before the allocation said to be deadlock avoidance.

Banker's Algorithm: It is the deadlock avoidance algorithm, the name was chosen because the bank never allocates more than the available cash.

Available: A vector of length 'm' indicates the number of available resources of each type. If $available[j]=k$, there are 'k' instances of resource types R_j available.

Allocation: An $n \times m$ matrix defines the number of resources of each type currently allocated to each process. If $allocation[i,j]=k$, then process P_i is currently allocated 'k' instances of resources type R_j .

Max: An $n \times m$ matrix defines the maximum demand of each process. If $max[i,j]=k$, then P_i may request at most 'k' instances of resource type R_j .

Need: An $n \times m$ matrix indicates the remaining resources need of each process. If $need[i,j]=k$, then P_i may need 'k' more instances of resource type R_j to complete this task. There fore, $Need[i,j]=Max[i,j]-Allocation[i,j]$

Safety Algorithm:

1. Work and Finish be the vector of length m and n respectively, $Work=Available$ and $Finish[i]=False$.

2. Find an i such that both

$Finish[i]=False$

$Need \leq Work$

If no such I exist go to step 4.

3. $Work=work + Allocation$, $Finish[i]=True$;

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

4. if $Finish[1]=True$ for all I, then the system is in safe state.

Resource request algorithm

Let Request i be request vector for the process P_i , If request $i=[j]=k$, then process P_i wants k instances of resource type R_j .

1. if $Request \leq Need$ go to step 2. Otherwise raise an error condition.
2. if $Request \leq Available$ go to step 3. Otherwise P_i must since the resources are available.
3. Have the system pretend to have allocated the requested resources to process P_i by modifying the state as follows;

$Available = Available - Request\ I;$

$Allocation\ I = Allocation + Request\ I;$

$Need\ i = Need\ i - Request\ I;$

If the resulting resource allocation state is safe, the transaction is completed and process P_i is allocated its resources. However, if the state is unsafe, the P_i must wait for Request i and the old resource-allocation state is restored.

Algorithm for Banker's Deadlock Avoidance:

1. Start the program.
2. Get the values of resources and processes.
3. Get the avail value.
4. After allocation find the need value.
5. Check whether its possible to allocate.
6. If it is possible then the system is in safe state.
7. Else system is not in safety state.
8. If the new request comes then check that the system is in safety.
9. or not if we allow the request.
10. stop the program.

/ Program to Simulate Bankers Algorithm for Dead Lock Avoidance */*

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
int alloc[10][10],max[10][10];
int avail[10],work[10],total[10];
int i,j,k,n,need[10][10];
int m;
int count=0,c=0;
char finish[10];
clrscr();
printf("Enter the no. of processes and resources:");
scanf("%d%d",&n,&m);
for(i=0;i<=n;i++)
finish[i]='\n';
printf("Enter the claim matrix:\n");
for(i=0;i<n;i++)
for(j=0;j<m;j++)
scanf("%d",&max[i][j]);
printf("Enter the allocation matrix:\n");
for(i=0;i<n;i++)
for(j=0;j<m;j++)
scanf("%d",&alloc[i][j]);
printf("Resource vector:");
for(i=0;i<m;i++)
scanf("%d",&total[i]);
for(i=0;i<m;i++)
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
for(i=0;i<n;i++)
for(j=0;j<m;j++)
avail[j]+=alloc[i][j];
for(i=0;i<m;i++)
work[i]=avail[i];
for(j=0;j<m;j++)
work[j]=total[j]-work[j];
for(i=0;i<n;i++)
for(j=0;j<m;j++)
need[i][j]=max[i][j]-alloc[i][j];
A:for(i=0;i<n;i++)
{
c=0;
for(j=0;j<m;j++)
if((need[i][j]<=work[j])&&(finish[i]=='n'))
c++;
if(c==m)
{
printf("All the resources can be allocated to Process %d",
i+1);
printf("\n\nAvailable resources are:");
for(k=0;k<m;k++)
{
work[k]+=alloc[i][k];
printf("%4d",work[k]);
}
printf("\n");
finish[i]='y';
printf("\nProcess %d executed?:%c \n",i+1,finish[i]);
count++;
}
}
if(count!=n)
goto A;
else
printf("\n System is in safe mode");
printf("\n The given state is safe state");
getch();
}
```

OUTPUT 1:

Enter the no. of processes and resources: 4 3

Enter the claim matrix:

```
3 2 2
6 1 3
3 1 4
4 2 2
```

Enter the allocation matrix:

```
1 0 0
6 1 2
2 1 1
0 0 2
```

Resource vector:9 3 6

All the resources can be allocated to Process 2

Available resources are: 6 2 3

Process 2 executed?:y

All the resources can be allocated to Process 3

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Available resources are: 8 3 4
Process 3 executed?:y
All the resources can be allocated to Process 4
Available resources are: 8 3 6
Process 4 executed?:y
All the resources can be allocated to Process 1
Available resources are: 9 3 6
Process 1 executed?:y
System is in safe mode
The given state is safe state

Experiment No:2 BANKER'S DEADLOCK PREVENTION

Aim: To Simulate Bankers Algorithm for Deadlock Prevention.

/* Program to Simulate Bankers Algorithm for Dead Lock Prevention */

```
#include<stdio.h>
#include<conio.h>
void main()
{
char job[10][10];
int time[10],avail,tem[10],temp[10];
int safe[10];
int ind=1,i,j,q,n,t;
clrscr();
printf("Enter no of jobs: ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter name and time: ");
scanf("%s%d",&job[i],&time[i]);
}
printf("Enter the available resources:");
scanf("%d",&avail);
for(i=0;i<n;i++)
{
temp[i]=time[i];
tem[i]=i;
}
for(i=0;i<n;i++)
for(j=i+1;j<n;j++)
{
if(temp[i]>temp[j])
{
t=temp[i];
temp[i]=temp[j];
temp[j]=t;
t=tem[i];
tem[i]=tem[j];
tem[j]=t;
}
}
for(i=0;i<n;i++)
{
q=tem[i];
if(time[q]<=avail)
{
```

Lab Manual

```
avail=avail-tem[q];
//printf("%s",job[safe[ind]]);
ind++;
}
else
{
printf("No safe sequence\n");
}
}
}
printf("Safe sequence is:");
for(i=1;i<ind; i++)
printf(" %s %d\n",job[safe[i]],time[safe[i]]);
getch();
}
```

OUTPUT 1:

```
Enter no of jobs:4
Enter name and time: A 1
Enter name and time: B 4
Enter name and time: C 2
Enter name and time: D 3
Enter the available resources: 20
Safe sequence is: A 1, C 2, D 3, B 4
```

Experiment No: 3 PAGING TECHNIQUE OF MEMORY MANAGEMENT

AIM:To implement the Memory management policy- Paging.

Description: Paging is an efficient memory management scheme because it is noncontiguous memory allocation method. The basic idea of paging is the physical memory (main memory) is divided into fixed sized blocks called frames, the logical address space is divided into fixed sized blocks, called pages, but page size and frame size should be equal. The size of the frame or a page is depending on operating system. In this scheme the operating system maintains a data structure that is page table; it is used for mapping purpose. The page table specifies the some useful information; it tells which frames are there and so on. The page table consisting of two fields, one is the page number and other one is frame number. Every address generated by the CPU divided into two parts; one is page number and second is page offset or displacement. The pages are loaded into available free frames in the physical memory.

Algorithm for Paging Technique:

- Step 1: Read all the necessary input from the keyboard.
- Step 2: Pages - Logical memory is broken into fixed - sized blocks.
- Step 3: Frames – Physical memory is broken into fixed – sized blocks.
- Step 4: Calculate the physical address using the following
Physical address = (Frame number * Frame size) + offset
- Step 5: Display the physical address.
- Step 6: Stop the process.

```
/* Program to simulate paging technique of memory management */
```

```
#include<stdio.h>
void main()
{
int p, ps, i;
int *sa;
clrscr();
printf("Enter how many pages: ");
scanf("%d",&np);
printf("Enter page size: ");
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
scanf("%d",&ps);
for(i=0;i< np;i++)
{
sa[i]=(int)malloc(ps);
printf("Page %d address is %d\n", i, sa[i]);
}
getch();
}
```

OUTPUT 1:

```
Enter how many pages: 5
Enter page size: 4
Page 0 address is 3080
Page 1 address is 3088
Page 2 address is 3096
Page 3 address is 3104
Page 4 address is 3112
```

OUTPUT 2:

```
Enter how many pages: 7
Enter page size: 8
Page 0 address is 3080
Page 1 address is 3096
Page 2 address is 3112
Page 3 address is 3128
Page 4 address is 3144
Page 5 address is 3160
Page 6 address is 3176
```

Experiment No: 4(a) THREAD CREATION

Aim: Design ,develop and execute a program using any thread library to create number of threads specified by the user ,each thread independently generate a random integer as an upper limit and then computes and prints the number of primes less than or equal to that upper limit along with that upper limit.

DESCRIPTION:

What is a Thread?

- Technically, a thread is defined as an independent stream of instructions that can be scheduled to run as such by the operating system. But what does this mean?
- To the software developer, the concept of a "procedure" that runs independently from its main program may best describe a thread.

What are Pthreads?

- Pthreads are defined as a set of C language programming types and procedure calls, implemented with a pthread.h header/include file and a thread library - though this library may be part of another library, such as lib, in some implementations.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Why Pthreads?

- In the world of high performance computing, the primary motivation for using Pthreads is to realize potential program performance gains.
- When compared to the cost of creating and managing a process, a thread can be created with much less operating system overhead. Managing threads requires fewer system resources than managing processes.
- All threads within a process share the same address space. Inter-thread communication is more efficient and in many cases, easier to use than inter-process communication.

The Pthreads API:

The original Pthreads API was defined in the ANSI/IEEE POSIX 1003.1 - 1995 standard. The POSIX standard has continued to evolve and undergo revisions, including the Pthreads specification. The subroutines which comprise the Pthreads API can be informally grouped into four major groups:

1. **Thread management:** Routines that work directly on threads - creating, detaching, joining, etc. They also include functions to set/query thread attributes (joinable, scheduling etc.)
2. **Mutexes:** Routines that deal with synchronization, called a "mutex", which is an abbreviation for "mutual exclusion". Mutex functions provide for creating, destroying, locking and unlocking mutexes.
3. **Condition variables:** Routines that address communications between threads that share a mutex. Based upon programmer specified conditions. Functions to set/query condition variable attributes are also included.
4. **Synchronization:** Routines that manage read/write locks and barriers.

Compiling Threaded Programs:

- Several examples of compile commands used for pthreads codes are listed in the table.

Compiler / Platform	Compiler Command	Description
INTEL Linux	icc -pthread	C
	icpc -pthread	C++
PGI Linux	pgcc -lpthread	C
	pgCC -lpthread	C++
GNU	gcc -pthread	GNU C

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Linux, Blue Gene	g++ -pthread	GNU C++
------------------	---------------------	---------

Thread management:

pthread_create(thread, attr, start routine, rag):

- Initially, your main () program comprises a single, default thread. All other threads must be explicitly created by the programmer.
- Pthread_create creates a new thread and makes it executable. This routine can be called any number of times from anywhere within your code.
- pthread_create arguments:
 - ✓ Thread: An opaque, unique identifier for the new thread returned by the subroutine.
 - ✓ Attr: An opaque attribute object that may be used to set thread attributes. You can specify a thread attributes object, or NULL for the default values.
 - ✓ start_routine: the C routine that the thread will execute once it is created.
 - ✓ Arg: A single argument that may be passed to *start_routine*. It must be passed by reference as a pointer cast of type void. NULL may be used if no argument is to be passed.

pthread_exit (): There are several ways in which a thread may be terminated:

- The thread returns normally from its starting routine. Its work is done.
- The thread makes a call to the pthread_exit subroutine - whether its work is done or not.
- The entire process is terminated due to making a call to either the exec() or exit()
- If main() finishes first, without calling pthread_exit explicitly itself.

Joining: pthread_join (threadid, status)

"Joining" is one way to accomplish synchronization between threads.

- The pthread_join () subroutine blocks the calling thread until the specified threadid thread terminates.
- The programmer is able to obtain the target thread's termination return status if it was specified in the target thread's call to pthread_exit ().
- A joining thread can match one pthread_join () call. It is a logical error to attempt multiple joins on the same thread.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Detaching: The pthread_detach () routine can be used to explicitly detach a thread even though it was created as joinable.

pthread_self (): Pthread_self returns the unique, system assigned thread ID of the calling thread.

PROGRAM CODE:

```
#include<stdio.h>
#include<pthread.h>
#include<math.h>
#include<stdlib.h>
Pthread_t ntip [50];

int isprime (int n)
{
    int i,cnt=0;
    if(n==2)
    {
        printf("\n %d ",n);
        return 0;
    }
    for(i=2;i<n;i++)
    {
        if(n%i==0)
            return 1;
    }
    printf("\n%d",n);
    return 0;
}

void prime_disp(int r)
{
    int i,cnt=0;
    printf(" \n In thread %u \n",(unsigned int)pthread_self());
    printf("\n Prime number list :\n");
    for(i=2;i<=r;i++)
    {
        if(!isprime(i))
            cnt++;
    }
    printf("\n no of prime number in the limit is %d\n",cnt);
}

void * thr_fn()
{
    printf("thread.....%u\n",(unsigned int)pthread_self());
    int r=rand();
    //printf(" \n In thread %d \n",(unsigned int)pthread_self());
    printf("Random number obtained = %d\n",r%50);
    prime_disp(r%50);
    return((void *)1);
}
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
}

int main()
{
    int err,i,n;
    void *tret;
    printf("Enter number of thread(upto 50) : ");
    scanf("%d",&n);
    for(i=0;i<n;i++) {
        err=pthread_create(&ntip[i],NULL,thr_fn,NULL);
        if(err!=0)
        {
            printf("Error");
            exit(0);
        }
    }
    //sleep(5);
    for(i=0;i<n;i++)
        pthread_join(ntip[i],&tret);
    printf("\n main thread\n ");
    return 0;
}
```

Output:

root@localhost student]# gcc -lpthread pp1.c

root@localhost student]# ./a.out

Enter number of thread(upto 50) : 3

Thread....2626696960

random no obtained=33

in thread 2626696960

prime no list:

2thread....2618304256

random no obtained=36

in thread 2618304256

prime no list:

2 3 5 7 11 13 17 19 23 29 31

no of prime no in the limit is 11

thread....2609911552

random no obtained=27

in thread 2609911552

prime no list:

3 5 7 11 13 17 19 23

No of prime no in the limit are 9

3 5 7 11 13 17 19 23 29 31

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment No- 4(b) PROCESS CREATION

Aim: Rewrite Experiment-4(a) such that the processes instead of threads are created and the number of child processes created is fixed as two. The program should make use of kernel timer to measure and print the real time, processor time user space time and kernel space for each process

DESCRIPTION:

Process is a program in execution. Process is a passive entity with a program counter specifying the next instruction to execute and a set of associated resources. Each process is represented in the operating system by a **process control block (PCB)** also called a **task control block**. It contains information such as process state, program counter, CPU registers.

System calls:

It provides the interface between a process and the operating system. It is a routine built in the kernel to perform function that requires communication with the system's hardware. All activities related to file handling; process and memory management and maintenance of user and system information are handled by the kernel using these system calls.

Kernel:

A Kernel timer is a data structure that instructs the kernel to execute a user-defined function with a user-defined argument at a user-defined time.

In computing, the kernel is the main component of most computer operating systems; it is a bridge between applications and the actual data processing done at the hardware level. The kernel's responsibilities include managing the system's resources (the communication between hardware and software components).

This program should use kernel timer to measure the following.

- 1) Real time.
- 2) User space time.
- 3) Kernel space time.
- 4) Processor time.

For both of the child processes as well as for parent process i.e., main ().

- ❖ **Processor time:** The amount of time a process takes to run, given that it has exclusive and uninterrupted use of the CPU. Note that in a modern computer, this would be very unusual, and so the processor time calculation for most processes involves adding up all the small amounts of time the CPU actually spends on the process. Some systems break processor time down into user time and system time.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

- ❖ **Real time**: Real time is the amount of time spent between process creation upto the end of process execution is considered.
- ❖ **User space time**: User space is that portion of system memory in which user processes run. The amount of time spent by a process in user mode from the time of creation to its process termination is called user space time.
- ❖ **Kernel space time**: The kernel is a program that constitutes the central core of a computer operating system. It is not a process, but rather a controller of processes, and it has complete control over everything that occurs on the system. This includes managing individual user processes within user space and preventing them from interfering with each other.

Keywords, Functions & Header Files used:

1) Times () - get process times

The times () function stores the current process times in the structtms that buf points to. The structtms is as defined in <sys/times.h>:

Structtms

```
{  
  
clock_t tms_utime; /* user time */  
  
clock_t tms_stime; /* system time */  
  
clock_t tms_cutime; /* user time of children */  
  
clock_t tms_cstime; /* system time of children */  
  
};
```

The tms_utime field contains the CPU time spent executing instructions of the calling process. The tms_stime field contains the CPU time spent in the system while executing tasks on behalf of the calling process. The tms_cutime field contains the sum of the tms_utime and tms_cutime values for all waited-for terminated children. The tms_cstime field contains the sum of the tms_stime and tms_cstime values for all waited-for terminated children.

Times for terminated children (and their descendants) are added in at the moment wait (2) or waitpid (2) returns their process ID. In particular, times of grandchildren that the children did not wait for are never seen. All times reported are in clock ticks.

Return Value: The function times returns the number of clock ticks that have elapsed since an arbitrary point in the past. For Linux this point is the moment the system was booted. This return

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

value may overflow the possible range of type `clock_t`. On error, `(clock_t) -1` is returned, and `errno` is set appropriately.

2) Wait ():- wait for process termination.

The `wait` function suspends execution of the current process until a child has exited, or until a signal is delivered whose action is to terminate the current process or to call a signal handling function. If a child has already exited by the time of the call (a so-called "zombie" process), the function returns immediately. Any system resources used by the child are freed.

3) Fork ():- create a child process.

`Fork` creates a child process that differs from the parent process only in its PID and PPID, and in the fact that resource utilizations are set to 0. File locks and pending signals are not inherited. Under Linux, `fork` is implemented using copy-on-write pages, so the only penalty incurred by `fork` is the time and memory required to duplicate the parent's page tables, and to create a unique task structure for the child.

Return Value: On success, the PID of the child process is returned in the parent's thread of execution, and a 0 is returned in the child's thread of execution. On failure, a -1 will be returned in the parent's context, no child process will be created, and `errno` will be set appropriately.

4) Exit ():- cause normal program termination.

The `exit ()` function causes normal program termination and the value of `status & 0377` is returned to the parent. All functions registered with `at exit ()` and `on exit ()` are called in the reverse order of their registration, and all open streams are flushed and closed. Files created by `tmpfile ()` are removed.

The C standard specifies two defines `EXIT_SUCCESS` and `EXIT_FAILURE` that may be passed to `toexit ()` to indicate successful or unsuccessful termination, respectively.

Return Value: The `exit ()` function does not return.

5) Unistd.h: This header file is used for system call wrapper functions such as `asfork ()`, `pipe ()` and I/O primitives such as `read`, `write`, `close`, etc.

PROGRAM CODE:

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
#include <unistd.h>

#include <sys/times.h>

child_process(int process_no)
{
    int i, n, prime, TotPrimeNo=0;
    struct tms chpr_t1, chpr_t2;
    long rchprt1, rchprt2;
    int Rand;

    rchprt1 = times(&chpr_t1); //start time of child process
    printf("\n Processor time for child process (%d) = %d", (process_no +1), rchprt1);
    printf("\n Created child process %d ", process_no+1);
    srand(process_no);
    Rand = random() % 10000;
    printf(" \n Random number = %d\t", Rand );
    // total prime no generation
    for(n=0; n<Rand; n++)
    {
        prime = 1; // initialize
        for(i=2; i<n; i++)
        {
            if(n % i == 0)
            {
                prime = 0;
                break;
            }
        }
        if(prime)
        {
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
    }
}

printf("\nTotal Prime Nos=%d\n", TotPrimeNo);

rchprt2 = times(&chpr_t2); // end time of child process

printf("----- The real time, user space and kernel space for chid process are:
-----");

printf("\n Real time for child process (%d) = %d", (process_no +1),
      (rchprt2- rchprt1));

printf("\n User space time for child process(%d) = %d", (process_no +1),
      (chpr_t2.tms_cutime -chpr_t1.tms_cutime));

printf("\n kernel space time for child process(%d) = %d\n\n", (process_no +1),
      (chpr_t2.tms_cstime – chpr_t1.tms_cstime));

exit(0);
}

void CreateProcess()
{
    int i;
    for (i=0; i<2;i++)
    {
        if(fork( ) == 0)
            child_process(i);
    }
    for (i=0; i<2;i++)
    {
        wait( );
    }
}

int main ()
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
{  
    long p_mt1, p_mt2;  
    struct tms p_mst1, p_mst2  
    p_mt1 = times(&p_mst1); //start time of a parent process  
    CreateProcess();  
    p_mt2 = times(&p_mst2); // end time of a parent process  
    printf("----- The real time, user space and kernel space for parent process are: -----  
/n");  
    printf("\n Real time for parent process = %d", p_mt2-p_mt1);  
    printf("\n User space time for parent process = %d", (p_mst2.tms_utime -  
                                                    p_mst1.tms_utime));  
    printf("\n Kernel space time for parent process = %d\n\n", (p_mst2.tms_stime -  
                                                    p_mst1.tms_stime));  
}
```

Output

```
[student@localhost ~]$ gcc -lpthread pp2.c
```

```
[student@localhost ~]$ ./a.out
```

```
Process Id.....4307
```

```
Random Number : 33
```

```
no of prime number 11
```

```
real time:81
```

```
cpu time:-998520
```

```
user time:-998520
```

```
kernal time:-998530
```

```
parentProcess Id.....4308
```

```
Random Number : 33
```

```
no of prime number 11
```

```
real time:62
```

user time:-998520

kernal time:-998530

Experiment No-5 Producer and Consumer Problem

Aim: Design, develop and implement a process with a producer thread and a consumer thread which make use of bounded buffer (size can be prefixed at a suitable value) for communication. Use any suitable synchronization construct.

DESCRIPTION:

The “Bounded-Buffer (Producer-Consumer) problem” is to create producer thread and consumer thread allows process to communicate and to synchronize their actions with the use of a bounded buffer. The bounded-buffer problem is a paradigm for cooperating processes, where the *producer* process produces information that is consumed by a *consumer* process.

The producer and the consumer, who share a common, fixed-size buffer. The producer's job is to generate a piece of data, put it into the buffer and start again. At the same time the consumer is consuming the data (i.e., removing it from the buffer) one piece at a time. The problem is to make sure that the producer won't try to add data into the buffer if it's full and that the consumer won't try to remove data from an empty buffer.

The key idea behind synchronization technique is to maintain data consistency, orderly execution of cooperating processes and preventing race condition.

Statement of the Problem:

The project entitled “Inter process Communication” has been designed using bounded-buffer problem to provide better paradigm for cooperating processes compare to other problems.

Producer-consumer problem (also known as the bounded-buffer problem) is a classical example of a multiprocessor synchronization problem. The problem describes two processes, the producer and the consumer, who share a common, fixed-size buffer. The producer's job is to generate a piece of data, put it into the buffer and start again. At the same time the consumer is consuming the data (i.e., removing it from the buffer) one piece at a time. The problem is to make sure that the producer won't try to add data into the buffer if it's full and that the consumer won't try to remove data from an empty buffer.

DESIGN:

This program can be divided mainly into three parts Producer, Consumer, Bounded-Buffer.

Producer: Producer corresponds to produce the item and stores in the buffer. After storing the item in the buffer, it has to increment the counter also it has to notice the consumer to consume the item.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Consumer consumes the item: Consumer corresponds to consume the item available in the buffer. After consuming the item in the buffer, it has to decrement the counter also it has to notice the consumer to consume the item.

Bounded-Buffer: It works as the interface between the Producer and the Consumer. It allows the producer to store the item in the buffer (shared space), also allows the consumer to consume the item from the buffer (shared space).

Producer-Consumer Problem:

Producer-consumer problem (also known as the bounded-buffer problem) is a classical example of a multiprocessor synchronization problem. The problem describes two processes, the producer and the consumer, who share a common, fixed-size buffer. The producer's job is to generate a piece of data, put it into the buffer and start again. At the same time the consumer is consuming the data (i.e., removing it from the buffer) one piece at a time. The problem is to make sure that the producer won't try to add data into the buffer if it's full and that the consumer won't try to remove data from an empty buffer.

The solution for the producer is to either go to sleep or discard data if the buffer is full. The next time the consumer removes an item from the buffer, it notifies the producer who starts to fill the buffer again. In the same way, the consumer can go to sleep if it finds the buffer to be empty. The next time the producer puts data into the buffer, it wakes up the sleeping consumer. The solution can be reached by means of inter-process communication, typically using semaphores. An inadequate solution could result in a deadlock where both processes are waiting to be awakened. The problem can also be generalized to have multiple producers and consumers.

Producer-Consumer problem is a paradigm for cooperating processes; *producer* process produces information that *consumer* process consumes the information.

- ❖ Unbounded-buffer places no practical limit on the size of the buffer.
- ❖ Bounded-buffer assumes that there is a fixed buffer size.

FLOW DIAGRAM:

Lab Manual

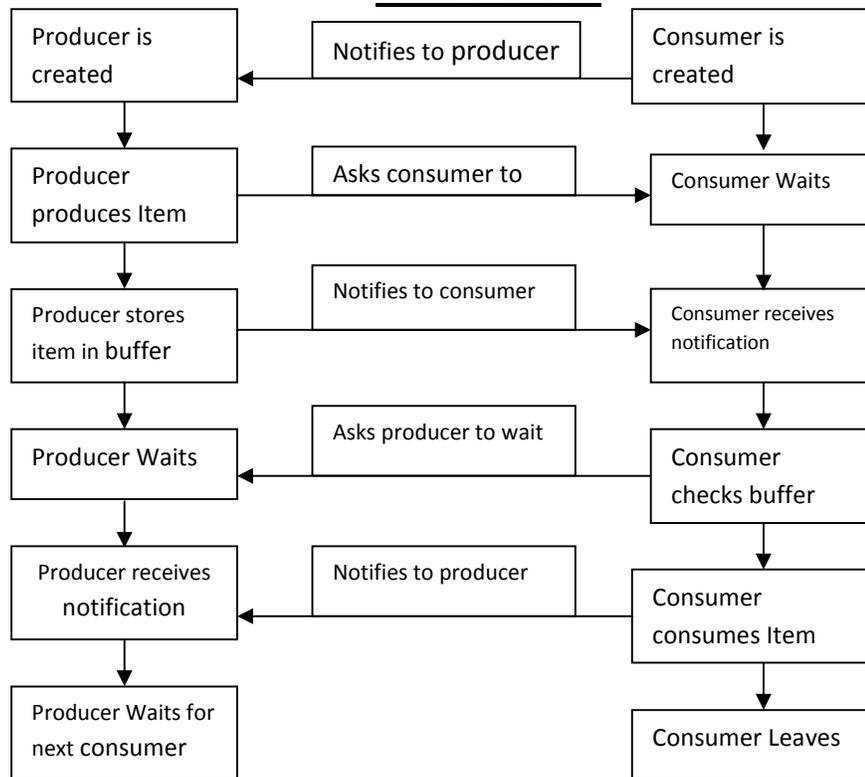


Figure 1.1: Flow diagram of Producer Consumer Problem

IMPLEMENTATION:

Algorithm

Producer process:

```
//item nextProduced;  
while (buffer_size == max_size) {  
    /* do nothing */  
    wait();  
}
```

Add item into buffer;

notify();

Consumer process:

```
//item to be Consumed;  
while (buffer_size == 0) {  
    /* do nothing */  
    wait();  
}
```

Remove item from buffer;

notify();

PROGRAM CODE:

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
/* buffer.h */
typedef int buffer_item;
#define BUFFER_SIZE 5

/* main.c */

#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include "buffer.h"
#define RAND_DIVISOR 100000000
#define TRUE 1

/* The mutex lock */
pthread_mutex_t mutex;
/* the semaphores */
sem_t full, empty;
/* the buffer */
buffer_item buffer[BUFFER_SIZE];
/* buffer counter */
int counter;

pthread_t tid;    //Thread ID
pthread_attr_t attr; //Set of thread attributes

void *producer(void *param); /* the producer thread */
void *consumer(void *param); /* the consumer thread */

void initializeData() {
    /* Create the mutex lock */
    pthread_mutex_init(&mutex, NULL);

    /* Create the full semaphore and initialize to 0 */
    sem_init(&full, 0, 0);

    /* Create the empty semaphore and initialize to BUFFER_SIZE */
    sem_init(&empty, 0, BUFFER_SIZE);

    /* Get the default attributes */
    pthread_attr_init(&attr);

    /* init buffer */
    counter = 0;
}

/* Producer Thread */
void *producer(void *param) {
    buffer_item item;

    while(TRUE) {
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
int rNum = rand() / RAND_DIVISOR;
sleep(rNum);

/* generate a random number */
item = rand();

/* acquire the empty lock */
sem_wait(&empty);
/* acquire the mutex lock */
pthread_mutex_lock(&mutex);

if(insert_item(item)) {
    fprintf(stderr, "Producer report error condition\n");
}

else {
    printf("producer produced %d\n", item);
}

/* release the mutex lock */
pthread_mutex_unlock(&mutex);
/* signal full */
sem_post(&full);
}

}

/* Consumer Thread */
void *consumer(void *param) {
    buffer_item item;

    while(TRUE) {
        /* sleep for a random period of time */
        int rNum = rand() / RAND_DIVISOR;
        sleep(rNum);

        /* acquire the full lock */
        sem_wait(&full);
        /* acquire the mutex lock */
        pthread_mutex_lock(&mutex);
        if(remove_item(&item)) {
            fprintf(stderr, "Consumer report error condition\n");
        }

        else {
            printf("consumer consumed %d\n", item);
        }

        /* release the mutex lock */
        pthread_mutex_unlock(&mutex);
        /* signal empty */
        sem_post(&empty);
    }
}

/* Add an item to the buffer */
int insert_item(buffer_item item) {
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
    /* When the buffer is not full add the item
    and increment the counter*/
    if(counter < BUFFER_SIZE) {
        buffer[counter] = item;
        counter++;
        return 0;
    }
    else { /* Error the buffer is full */
        return -1;
    }
}

/* Remove an item from the buffer */
int remove_item(buffer_item *item) {
    /* When the buffer is not empty remove the item
    and decrement the counter */
    if(counter > 0) {
        *item = buffer[(counter-1)];
        counter--;
        return 0;
    }
    else { /* Error program code

/* buffer.h */
typedef int buffer_item;
#define BUFFER_SIZE 5

/* main.c */

#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include "buffer.h"

#define RAND_DIVISOR 100000000
#define TRUE 1

int main(int argc, char *argv[]) {
    /* Loop counter */
    int i;

    /* Verify the correct number of arguments were passed in */
    if(argc != 4) {
        fprintf(stderr, "USAGE:./main.out <INT> <INT> <INT>\n");
    }

    int mainSleepTime = atoi(argv[1]); /* Time in seconds for main to sleep */
    int numProd = atoi(argv[2]); /* Number of producer threads */
    int numCons = atoi(argv[3]); /* Number of consumer threads */
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
/* Initialize the app */
initializeData();

/* Create the producer threads */
for(i = 0; i < numProd; i++) {
    /* Create the thread */
    pthread_create(&tid,&attr,producer,NULL);
}

/* Create the consumer threads */
for(i = 0; i < numCons; i++) {
    /* Create the thread */
    pthread_create(&tid,&attr,consumer,NULL);
}

/* Sleep for the specified amount of time in milliseconds */
sleep(mainSleepTime);

/* Exit the program */
printf("Exit the program\n");
exit(0);
}

/* The mutex lock */
pthread_mutex_t mutex;

/* the semaphores */
sem_t full, empty;

/* the buffer */
buffer_item buffer[BUFFER_SIZE];

/* buffer counter */
int counter;

pthread_t tid;    //Thread ID
pthread_attr_t attr; //Set of thread attributes

void *producer(void *param); /* the producer thread */
void *consumer(void *param); /* the consumer thread */

void initializeData() {

    /* Create the mutex lock */
    pthread_mutex_init(&mutex, NULL);

    /* Create the full semaphore and initialize to 0 */
    sem_init(&full, 0, 0);

    /* Create the empty semaphore and initialize to BUFFER_SIZE */
    sem_init(&empty, 0, BUFFER_SIZE);
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
/* Get the default attributes */
pthread_attr_init(&attr);

int main(int argc, char *argv[]) {
    /* Loop counter */
    int i;

    /* Verify the correct number of arguments were passed in */
    if(argc != 4) {
        fprintf(stderr, "USAGE: ./main.out <INT> <INT> <INT>\n");
    }

    int mainSleepTime = atoi(argv[1]); /* Time in seconds for main to sleep */
    int numProd = atoi(argv[2]); /* Number of producer threads */
    int numCons = atoi(argv[3]); /* Number of consumer threads */

    /* Initialize the app */
    initializeData();

    /* Create the producer threads */
    for(i = 0; i < numProd; i++) {
        /* Create the thread */
        pthread_create(&tid,&attr,producer,NULL);
    }

    /* Create the consumer threads */
    for(i = 0; i < numCons; i++) {
        /* Create the thread */
        pthread_create(&tid,&attr,consumer,NULL);
    }

    /* Sleep for the specified amount of time in milliseconds */
    sleep(mainSleepTime);

    /* Exit the program */
    printf("Exit the program\n");
    exit(0);
}

/* Producer Thread */
void *producer(void *param) {
    buffer_item item;

    while(TRUE) {
        /* sleep for a random period of time */
        int rNum = rand() / RAND_DIVISOR;
        sleep(rNum);

        /* generate a random number */
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
/* acquire the empty lock */
sem_wait(&empty);
/* acquire the mutex lock */
pthread_mutex_lock(&mutex);

if(insert_item(item)) {
    fprintf(stderr, "Producer report error condition\n");
}
else {
    printf("producer produced %d\n", item);
}
/* release the mutex lock */
pthread_mutex_unlock(&mutex);
/* signal full */
sem_post(&full);
}
}
```

```
/* Consumer Thread */
void *consumer(void *param) {
    buffer_item item;

    while(TRUE) {
        /* sleep for a random period of time */
        int rNum = rand() / RAND_DIVISOR;
        sleep(rNum);

        /* acquire the full lock */
        sem_wait(&full);
        /* acquire the mutex lock */
        pthread_mutex_lock(&mutex);
        if(remove_item(&item)) {
            fprintf(stderr, "Consumer report error condition\n");
        }
        else {
            printf("consumer consumed %d\n", item);
        }
        /* release the mutex lock */
        pthread_mutex_unlock(&mutex);
        /* signal empty */
        sem_post(&empty);
    }
}
```

```
/* Add an item to the buffer */
int insert_item(buffer_item item) {
    /* When the buffer is not full add the item
    and increment the counter*/
    if(counter < BUFFER_SIZE) {
        buffer[counter] = item;
        counter++;
    }
}
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
    return 0;
}
else { /* Error the buffer is full */
    return -1;
}
}

/* Remove an item from the buffer */
int remove_item(buffer_item *item) {
    /* When the buffer is not empty remove the item
    and decrement the counter */
    if(counter > 0) {
        *item = buffer[(counter-1)];
        counter--;
        return 0;
    }
    else { /* Error buffer empty */
        return -1;
    }
}

int main(int argc, char *argv[]) {
    /* Loop counter */
    int i;

    /* Verify the correct number of arguments were passed in */
    if(argc != 4) {
        fprintf(stderr, "USAGE: ./main.out <INT> <INT> <INT>\n");
    }

    int mainSleepTime = atoi(argv[1]); /* Time in seconds for main to sleep */
    int numProd = atoi(argv[2]); /* Number of producer threads */
    int numCons = atoi(argv[3]); /* Number of consumer threads */

    /* Initialize the app */
    initializeData();

    /* Create the producer threads */
    for(i = 0; i < numProd; i++) {
        /* Create the thread */
        pthread_create(&tid,&attr,producer,NULL);
    }

    /* Create the consumer threads */
    for(i = 0; i < numCons; i++) {
        /* Create the thread */
        pthread_create(&tid,&attr,consumer,NULL);
    }

    /* Sleep for the specified amount of time in milliseconds */
    sleep(mainSleepTime);
}
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
/* Exit the program */  
printf("Exit the program\n");  
exit(0);  
}
```

OUTPUT :

```
File Edit View Search Terminal Help  
[student@localhost ~]$ gcc -lpthread pro3.c  
[student@localhost ~]$ ./a.out 10 10 10  
producer produced 35005211  
consumer consumed 35005211  
producer produced 1726956429  
consumer consumed 1726956429  
producer produced 278722862  
consumer consumed 278722862  
producer produced 468703135  
producer produced 1801979802  
producer produced 635723058  
producer produced 1125898167  
consumer consumed 1125898167  
Exit the program  
[student@localhost ~]$ □
```

Experiment No- 6 Remote Procedure Call

Aim: Design, develop, and execute a program to demonstrate the use of RPC.

DESCRIPTION:

Procedure Call (RPC) defines a powerful technology for creating distributed client/server programs. The RPC run-time stubs and libraries manage most of the processes relating to network protocols and communication. This enables you to focus on the details of the application rather than the details of the network.

Examples: File service, Database service, and Authentication service.

Need of RPC: The client needs an easy way to call the procedures of the server to get some services. RPC enables clients to communicate with servers by calling procedures in a similar way to the conventional use of procedure calls in high-level languages. RPC is modeled on the local procedure call, but the called procedure is executed in a different process and usually a different computer.

RMI Architecture:

RMI system allows an object running in one JVM to invoke methods on an object in another JVM. RMI can communicate only between programs written in Java.

RMI applications often consist of two applications: A sever and a client

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

The application is divided into objects. A server creates remote objects and registers these objects with a simple bootstrap name server called a registry (rmiregistry). The client gets a handle by looking up the remote object by its name in the server(s) registry and invokes methods on it.

Remote object :

A remote object is one whose methods can be invoked from another JVM practically on a different machine. In situations where you do not have a network, or a client, or a server available to test programs, you can invoke methods on a remote object from another JVM on the same machine. A remote object must implement at least one interface that extends the java.rmi.Remote interface.

Remote Interface :

A remote interface is an interface, which declares a set of methods that may be invoked remotely (from a different JVM). All interactions with the remote object will be performed through this interface. A remote interface must directly or indirectly extend the java.rmi.Remote interface. Each method declaration in a remote interface must include java.rmi.RemoteException or one of its super classes in its throws clause.

The Remote Procedure Call (RPC) message protocol consists of two distinct structures: the call message and the reply message.

RPC Call Message: Each remote procedure call message contains the following unsigned integer fields to uniquely identify the remote procedure:

- ✓ Program number
- ✓ Program version number
- ✓ Procedure number

The body of an RPC call message takes the following form:

```
struct call_body {  
  
    unsigned int rpcvers;  
  
    unsigned int prog;  
  
    unsigned int vers;  
  
    unsigned int proc;
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
opaque_auth verf;
```

```
1 parameter 2 parameter.
```

```
};
```

RPC Reply Message: The RPC protocol for a reply message varies depending on whether the call message is accepted or rejected by the network server. The reply message to a request contains information to distinguish the following conditions:

- RPC executed the call message successfully.
- The remote implementation of RPC is not protocol version 2. The lowest and highest supported RPC version numbers are returned.
- The remote program is not available on the remote system.
- The remote program does not support the requested version number. The lowest and highest supported remote program version numbers are returned.
- The requested procedure number does not exist. This is usually a caller-side protocol or programming error.

The RPC reply message takes the following form:

```
enum reply_stat stat {  
  
    MSG_ACCEPTED = 0,  
  
    MSG_DENIED = 1  
  
};
```

Function used:

- **XmlRpcClientConfigImpl():**

Default implementation of a clients request configuration.

- **xmlRpcServer():**

This is the embedded XML-RPC server, which is called to execute the clients requests Obviously, this is an extremely fast transport. However, its main use is for debugging and development.

PROGRAM CODE:

```
/**SERVER PROGRAM**//
```

```
import org.apache.xmlrpc.server.PropertyHandlerMapping;
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
import org.apache.xmlrpc.webserver.WebServer;

public class RpcServer {

    public Integer sum(int x, int y) {

        System.out.println("Finding sum for " + x + " and " + y);

        return new Integer(x + y);

    }

    public static void main(String[] args) {

        try {

            System.out.println("Attempting to start XML-RPC Server...");

            WebServer server = new WebServer(8080);

            PropertyHandlerMapping phm = new PropertyHandlerMapping();

            phm.addHandler("math", RpcServer.class);

            server.getXmlRpcServer().setHandlerMapping(phm);

            server.start();

            System.out.println("Started successfully.");

            System.out.println("Accepting requests. (Halt program to stop.)");

        }

        catch (Exception exception) {

            System.err.println("JavaServer: " + exception);

        }

    }

}
```

/CLIENT PROGRAM**/**

```
import java.net.URL;
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
import java.util.Vector;

import org.apache.xmlrpc.client.XmlRpcClient;

import org.apache.xmlrpc.client.XmlRpcClientConfigImpl;

public class RpcClient {

    public static void main(String[] args) throws Exception {

        XmlRpcClientConfigImpl config = new XmlRpcClientConfigImpl();

        String serverAddress = "localhost:8080";

        int val1 = 0;

        int val2 = 0;

        if (args.length > 0) {

            serverAddress = args[0];

        }

        if (args.length >= 2) {

            val1 = Integer.parseInt(args[1]);

            val2 = Integer.parseInt(args[2]);

        }

        config.setServerURL(new URL("http://" + serverAddress + "/RPC2"));

        XmlRpcClient server = new XmlRpcClient();

        server.setConfig(config);

        Vector<Integer> params = new Vector<Integer>();

        params.addElement(new Integer(val1));

        params.addElement(new Integer(val2));

        Object result = server.execute("math.sum", params);

        int sum = ((Integer) result).intValue();

        System.out.println("The sum is: " + sum);

    }

}
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR
Lab Manual

```
    }  
}
```

Output

Server:

Attempting to start XML-RPC Server...

Started successfully.

Accepting requests. (Halt program to stop.)

Finding sum for 17 and 13

Finding sum for 54 and 23

Client:

\$ java RpcClient localhost:8080 17 13

The sum is: 30

\$ java RpcClient localhost:8080 54 23

The sum is: 77

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Advanced Programming Lab

Course Code: MCSE192

L-T-P scheme: 0-0-3

Course Credit: 2

Objectives:

1. To learn and understand different types artificial neural network algorithm.
2. To learn MatLab for the programming of ANN.

Learning Outcomes: The students will have a detailed knowledge of the concepts of matlab. Upon the completion of Advanced algorithm course, the student will be able to:

- **Understand** and implement basic services and functionalities of the ANN using matlab.
- **Use** KohonenSelfOrganizingfeaturemaptoClusterthe vectorsusingowninitial weightsand learningrate.
- **Understand** the benefits of artificial neural network in artificial intelligence.

Course Contents:

Exercises that must be done in this course are listed below:

Exercise 1: Program to generate a few activation functions that are being used in neural network

Exercise 2: Program to classify with a 2-input perceptron.

Exercise 3: Program for perceptron net for an AND function with bipolar inputs and targets.

Exercise 4: Develop a Matlab program for OR function with bipolar inputs and targets using ADALINE network.

Exercise 5: Develop a Matlab program to generate XOR function for bipolar inputs and targets using MADALINE network.

Exercise 6: Develop a Matlab program to store the vector (-1,-1,-1,-1) and (-1,-1,1,1) in an auto-associative network. Find the weight matrix. Test the net with (1,1,1,1) as input.

Exercise 7: Consider a vector (1,0,1,1) to be stored in the net. Test a discrete Hopfield net with error in the 1st and 4th components (0,0,1,0) of the stored vector.

Exercise 8: Develop a Matlab program for XOR function (binary input and output) with momentum factor using back-propagation algorithm.

Exercise 9: Develop a Matlab program for drawing feature maps (Kohonen Self Organizing Feature maps) in 1-Dimensional view.

Exercise 10: Use Kohonen Self Organizing feature map to cluster the vectors (assume four binary vectors) using own initial weights (to be assumed) and learning rate (to be assumed).

Text Book:

1. S. N. Sivanandam, S. N. Deepa, Introduction to Neural Networks Using Matlab 6.0, 1st edition, Tata McGraw-Hill Education, 2006

References:

1. B. Yegnanarayana, Artificial Neural Networks, Prentice Hall of India, 1999
2. Satish Kumar, Neural Networks – A Classroom Approach, Tata McGraw-Hill, 2003
3. S. Haykin, Neural Networks – A Comprehensive Foundation, Prentice Hall, 1998
4. C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006

Recommended Systems/Software Requirements:

1. Intel based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
2. Matlab software in Windows XP .

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Exercise1: Generate a few activationfunctionthat are being used in neural network

Description:

Activation Function :

A function used to transform the activation level of a unit (neuron) into an output signal. Typically, activation functions have a "squashing" effect. Together with the PSP function (which is applied first) this defines the unit type. Neural Networks supports a wide range of activation functions. Only a few of these are used by default; the others are available for customization.

- Identity: The activation level is passed on directly as the output. Used in a variety of network types, including linear networks, and the output layer of radial basis function networks.
- Logistic: This is an S-shaped (sigmoid) curve, with output in the range (0,1).
- Hyperbolic: The hyperbolic tangent function (tanh): a sigmoid curve, like the logistic function, except that output lies in the range (-1,+1). Often performs better than the logistic function because of its symmetry. Ideal for customization of multilayer perceptrons, particularly the hidden layers.
- Exponential: The negative exponential function. Ideal for use with radial units. The combination of radial synaptic function and negative exponential activation function produces units that model a Gaussian (bell-shaped) function centred at the weight vector.
- Softmax: Exponential function, with results normalized so that the sum of activations across the layer is 1.0. Can be used in the output layer of multilayer perceptrons for classification problems, so that the outputs can be interpreted as probabilities of class membership (Bishop, 1995; Bridle, 1990).
- Unit sum: Normalizes the outputs to sum to 1.0. Used in PNNs to allow the outputs to be interpreted as probabilities.
- Square root: Used to transform the squared distance activation in an SOFM network or Cluster network to the actual distance as an output.
- Sine: Possibly useful if recognizing radially-distributed data; not used by default.
- Ramp: A piece-wise linear version of the sigmoid function. Relatively poor training performance, but fast execution.
- Step: Outputs either 1.0 or 0.0, depending on whether the Synaptic value is positive or negative. Can be used to model simple networks such as perceptrons.

Aim: Programto generate a few activationfunctionthat are being used in neural network

Program

```
% Illustration of various activation functions used in NN's
```

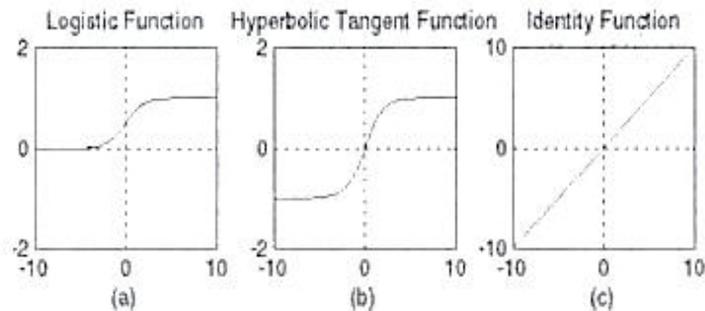
```
x = -10:0.1:10;  
tmp = exp(-x);  
y1 = 1./(1+tmp);  
y2 = (1-tmp)./(1+tmp);  
y3 = x;  
subplot(231); plot(x, y1); grid on;  
axis([min(x) max(x) -2 2]);  
title('Logistic Function');  
xlabel('a');  
axis('square');  
subplot(232); plot(x, y2); grid on;  
axis([min(x) max(x) -2 2]);
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
xlabel('b');  
axis('square');  
subplot(233); plot(x, y3); grid on;  
axis([min(x) max(x) min(x) max(x)]);  
title('Identity Function');  
xlabel('c');  
axis('square');
```

Output:



Exercise2: Program to classify with a 2-input perceptron.

Description:

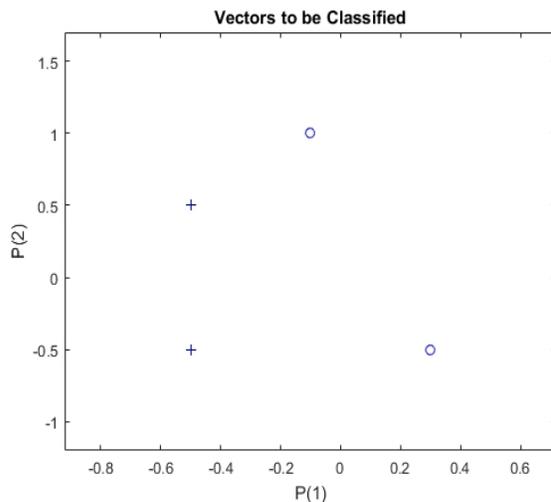
A 2-input hard limit neuron is trained to classify 5 input vectors into two categories.

Each of the five column vectors in X defines a 2-element input vectors and a row vector T defines the vector's target categories. We can plot these vectors with PLOTTPV.

Program(A):

```
X = [ -0.5 -0.5 +0.3 -0.1;  
      -0.5 +0.5 -0.5 +1.0];  
T = [1 1 0 0];  
plotpv(X,T);
```

output:



The perceptron must properly classify the 5 input vectors in X into the two categories defined by T. Perceptrons have HARDLIM neurons. These neurons are capable of separating an input space with a straight line into two categories (0 and 1).

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

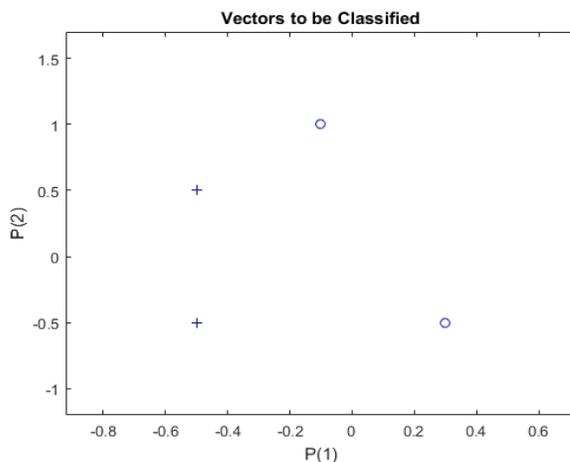
Here PERCEPTRON creates a new neural network with a single neuron. The network is then configured to the data, so we can examine its initial weight and bias values. (Normally the configuration step can be skipped as it is automatically done by ADAPT or TRAIN.)

```
net = perceptron;  
net = configure(net,X,T);
```

The input vectors are replotted with the neuron's initial attempt at classification.

The initial weights are set to zero, so any input gives the same output and the classification line does not even appear on the plot. Fear not... we are going to train it!

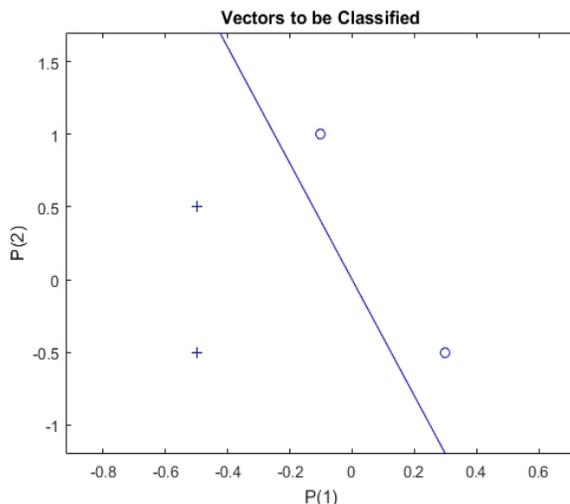
```
plotpv(X,T);  
plotpc(net.IW{1},net.b{1});
```



Here the input and target data are converted to sequential data (cell array where each column indicates a timestep) and copied three times to form the series XX and TT.

ADAPT updates the network for each timestep in the series and returns a new network object that performs as a better classifier.

```
XX = repmat(con2seq(X),1,3);  
TT = repmat(con2seq(T),1,3);  
net = adapt(net,XX,TT);  
plotpc(net.IW{1},net.b{1});
```

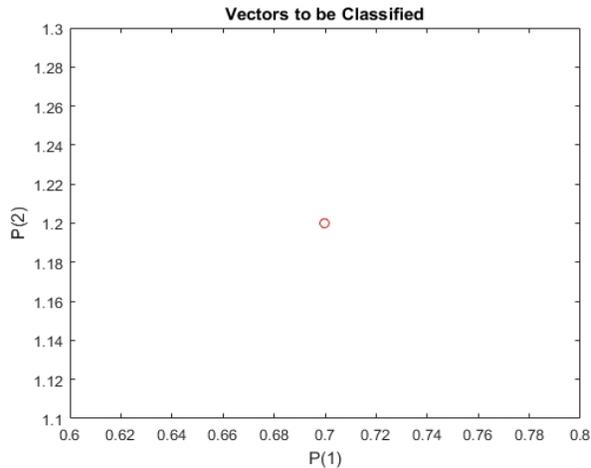


Now SIM is used to classify any other input vector, like [0.7; 1.2]. A plot of this new point with the original training set shows how the network performs. To distinguish it from the training set, color it red

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

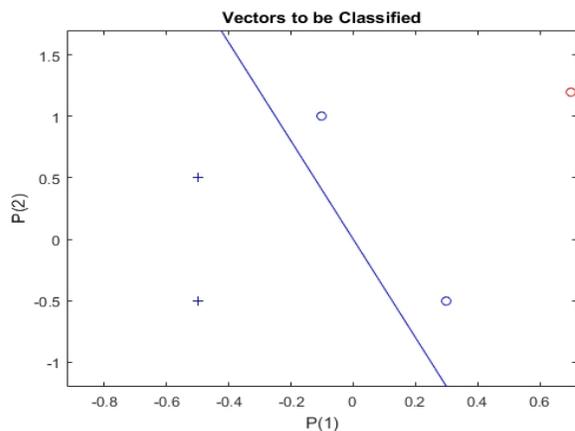
```
x = [0.7; 1.2];  
y = net(x);  
plotpv(x,y);  
point = findobj(gca,'type','line');  
point.Color = 'red';
```



Turn on "hold" so the previous plot is not erased and plot the training set and the classification line.

The perceptron correctly classified our new point (in red) as category "zero" (represented by a circle) and not a "one" (represented by a plus).

```
hold on;  
plotpv(X,T);  
plotpc(net.IW{1},net.b{1});  
hold off;
```



Exercise 3: Perceptron net for an AND function with bipolar inputs and targets.

Description:

A perceptron is an artificial neuron using the Heaviside step function as the activation function. The perceptron algorithm is also termed the single-layer perceptron, to distinguish it from a multilayer perceptron, which is a misnomer for a more complicated neural network. As a linear classifier, the single-layer perceptron is the simplest feed forward neural network.

Aim: Program for perceptron net for an AND function with bipolar inputs and targets.

The truth table for the AND function is given as

$$\begin{array}{ccc} X_1 & X_2 & Y \end{array}$$

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

- 1	1	- 1
1	- 1	- 1
1	1	1

The MATLAB program for the above table is given as follows.

Program

```
%Perceptron for AND funtion
clear;
clc;
x=[1 1 -1 -1;1 -1 1 -1];
t=[1 -1 -1 -1];
w=[0 0];
b=0;
alpha=input('Enter Learning rate=');
theta=input('Enter Threshold value=');
con=1;
epoch=0;
while con
    con=0;
    for i=1:4
        yin=b+x(1,i)*w(1)+x(2,i)*w(2);
        if yin>theta
            y=1;
        end
        if yin <=theta & yin>=-theta
            y=0;
        end
        if yin<-theta
            y=-1;
        end
        if y-t(i)
            con=1;
            for j=1:2
                w(j)=w(j)+alpha*t(i)*x(j,i);
            end
            b=b+alpha*t(i);
        end
    end
    epoch=epoch+1;
end
disp('Perceptron for AND funtion');
disp(' Final Weight matrix');
disp(w);
disp('Final Bias');
disp(b);
```

Output

```
Enter Learning rate=1
Enter Threshold value=0.5
Perceptron for AND funtion
Final Weight matrix
    1    1
Final Bias
    -1
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Exercise 4: OR function with bipolar inputs and targets using ADALINE network.

Description:

ADALINE (Adaptive Linear Neuron or later Adaptive Linear Element) is an early single-layer artificial neural network and the name of the physical device that implemented this network.

Aim: Write a Matlab program for OR function with bipolar inputs and targets using ADALINE network.

The truth table for the OR function with bipolar inputs and targets is given as,

X_1	X_2	Y
-1	-1	-1
-1	1	1
1	-1	1
1	1	1

Program:

```
clear all;
clc;
disp('Adaline network for OR function Bipolar inputs and targets');
%input pattern
x1=[1 1 -1 -1];
x2=[1 -1 1 -1];
%bias input
x3=[1 1 1 1];
%target vector
t=[1 1 1 -1];
%initial weights and bias
w1=0.1;w2=0.1;b=0.1;
%initialize learning rate
alpha=0.1;
%error convergence
e=2;
%change in weights and bias
delw1=0;delw2=0;delb=0;
epoch=0;
while(e>1.018)
    epoch=epoch+1;
    e=0;
    for i=1:4
        nety(i)=w1*x1(i)+w2*x2(i)+b;
        %net input calculated and target
        nt=[nety(i) t(i)];
        delw1=alpha*(t(i)-nety(i))*x1(i);
        delw2=alpha*(t(i)-nety(i))*x2(i);
        delb=alpha*(t(i)-nety(i))*x3(i);
        %weight changes
        wc=[delw1 delw2 delb]
        %updating of weights
        w1=w1+delw1;
        w2=w2+delw2;
        b=b+delb;
        %new weights
        w=[w1 w2 b]
        %input pattern
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```

%printing the results obtained
pnt=[x nt wc w]
end
for i=1:4
    nety(i)=w1*x1(i)+w2*x2(i)+b;
    e=e+(t(i)-nety(i))^2;
end
end
end

```

Output

	X1	X2	b	Netin	Target	delw1	delw2	delb	w1(n)	w2(n)	b(n)
Epoch 1	1.0000	1.0000	1.0000	0.3000	1.0000	0.0700	0.0700	0.0700	0.1700	0.1700	0.1700
	1.0000	-1.0000	1.0000	0.1700	1.0000	0.0830	-0.0830	0.0830	0.2530	0.0870	0.2530
	-1.0000	1.0000	1.0000	0.0870	1.0000	-0.0913	0.0913	0.0913	0.1617	0.1783	0.3443
	-1.0000	-1.0000	1.0000	0.0043	-1.0000	0.1004	0.1004	-0.1004	0.2621	0.2787	0.2439
Epoch 2	1.0000	1.0000	1.0000	0.7847	1.0000	0.0215	0.0215	0.0215	0.2837	0.3003	0.2654
	1.0000	-1.0000	1.0000	0.2488	1.0000	0.0751	-0.0751	0.0751	0.3588	0.2251	0.3405
	-1.0000	1.0000	1.0000	0.2069	1.0000	-0.0793	0.0793	0.0793	0.2795	0.3044	0.4198
	-1.0000	-1.0000	1.0000	-0.1641	-1.0000	0.0836	0.0836	-0.0836	0.3631	0.3880	0.3362
Epoch 3	1.0000	1.0000	1.0000	1.0873	1.0000	-0.0087	-0.0087	-0.0087	0.3543	0.3793	0.3275
	1.0000	-1.0000	1.0000	0.3025	1.0000	0.0697	-0.0697	0.0697	0.4241	0.3096	0.3973
	-1.0000	1.0000	1.0000	0.2827	1.0000	-0.0717	0.0717	0.0717	0.3523	0.3813	0.4690
	-1.0000	-1.0000	1.0000	-0.2647	-1.0000	0.0735	0.0735	-0.0735	0.4259	0.4548	0.3954
Epoch 4	1.0000	1.0000	1.0000	1.2761	1.0000	-0.0276	-0.0276	-0.0276	0.3983	0.4272	0.3678
	1.0000	-1.0000	1.0000	0.3389	1.0000	0.0661	-0.0661	0.0661	0.4644	0.3611	0.4339
	-1.0000	1.0000	1.0000	0.3307	1.0000	-0.0669	0.0669	0.0669	0.3974	0.4280	0.5009
	-1.0000	-1.0000	1.0000	-0.3246	-1.0000	0.0675	0.0675	-0.0675	0.4650	0.4956	0.4333
Epoch 5	1.0000	1.0000	1.0000	1.3939	1.0000	-0.0394	-0.0394	-0.0394	0.4256	0.4562	0.3939
	1.0000	-1.0000	1.0000	0.3634	1.0000	0.0637	-0.0637	0.0637	0.4893	0.3925	0.4576
	-1.0000	1.0000	1.0000	0.3609	1.0000	-0.0639	0.0639	0.0639	0.4253	0.4564	0.5215
	-1.0000	-1.0000	1.0000	-0.3603	-1.0000	0.0640	0.0640	-0.0640	0.4893	0.5204	0.4575

Exercise 5: XOR function for bipolar inputs and targets using MADALINE Network.

Description:

MADALINE (Many ADALINE^[1]) is a three-layer (input, hidden, output), fully connected, feed-forward artificial neural network architecture for classification that uses ADALINE units in its hidden and output layers, i.e. its activation function is the sign function.^[2] The three-layer network uses memistors. Three different training algorithms for MADALINE networks, which cannot be learned using backpropagation because the sign function is not differentiable, have been suggested, called Rule I, Rule II and Rule III. The first of these dates back to 1962 and cannot adapt the weights of the hidden-output connection.^[3] The second training algorithm improved on Rule I and was described in 1988.^[1] The third "Rule" applied to a modified network with sigmoid activations instead of signum; it was later found to be equivalent to backpropagation.^[3]

The Rule II training algorithm is based on a principle called "minimal disturbance". It proceeds by looping over training examples, then for each example, it:

- finds the hidden layer unit (ADALINE classifier) with the lowest confidence in its prediction,
- tentatively flips the sign of the unit,
- accepts or rejects the change based on whether the network's error is reduced

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Aim: Write a Matlab program to generate XOR function for bipolar inputs and targets using MADALINE Network.

The truth table for XOR function with bipolar inputs and targets is given as,

X_1	X_2	Y
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

Program

```
%Madaline for XOR function
clc;
clear;
%Input and Target
x=[1 1 -1 -1;1 -1 1 -1];
t=[-1 1 1 -1];
%Assume initial weight matrix and bias
w=[0.05 0.1;0.2 0.2];
b1=[0.3 0.15];
v=[0.5 0.5];
b2=0.5;
con=1;
alpha=0.5;
epoch=0;
while con
    con=0;
    for i=1:4
        for j=1:2
            zin(j)=b1(j)+x(1,i)*w(1,j)+x(2,i)*w(2,j);
            if zin(j)>=0
                z(j)=1;
            else
                z(j)=-1;
            end
        end
        yin=b2+z(1)*v(1)+z(2)*v(2);
        if yin>=0
            y=1;
        else
            y=-1;
        end
        if y~=t(i)
            con=1;
            if t(i)==1
                if abs(zin(1)) > abs(zin(2))
                    k=2;
                else
                    k=1;
                end
                b1(k)=b1(k)+alpha*(1-zin(k));
                w(1:2,k)=w(1:2,k)+alpha*(1-zin(k))*x(1:2,i);
            else
                for k=1:2
                    if zin(k)>0

```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
        w(1:2,k)=w(1:2,k)+alpha*(-1-zin(k))*x(1:2,i);
    end
end
end
end
epoch=epoch+1;
end
disp('Weight matrix of hidden layer');
disp(w);
disp('Bias of hidden layer');
disp(b1);
disp('Total Epoch');
disp(epoch);
```

Output:

```
Weight matrix of hidden layer
 1.3203 -1.2922
-1.3391  1.2859
Bias of hidden layer
-1.0672 -1.0766
Total Epoch
 3
```

Exercise 6: Store the vector (-1,-1,-1,-1) and (-1,-1,1,1) in an auto-associative network. Find the weight matrix. Test the net with (1,1,1,1) as input.

Description:

Auto-associative networks are a special subset of the hetero-associative networks (Hetero-associative networks map m input vectors x_1, x_2, \dots, x_m in n -dimensional space to m output vectors y_1, y_2, \dots, y_m in k -dimensional space, so that $x_i \rightarrow y_i$. If $\|x - x_i\|^2 < \epsilon$ then $x \rightarrow y_i$. This should be achieved by the learning algorithm, but becomes very hard when the number m of vectors to be learned is too high.), in which each vector is associated with itself, i.e., $y_i = x_i$ for $i = 1, \dots, m$. The function of such networks is to correct noisy input vectors.

Aim: Write a Matlab program to store the vector (-1,-1,-1,-1) and (-1,-1,1,1) in an auto-associative network. Find the weight matrix. Test the net with (1,1,1,1) as input.

Program

```
clc;
clear;
x=[-1 -1 -1 -1;-1 -1 1 1];
t=[1 1 1 1];
w=zeros(4,4);
for i=1:2
    w=w+x(i,1:4)'*x(i,1:4);
end
yin=t*w;
for i=1:4
    if yin(i)>0
        y(i)=1;
    else
        y(i)=-1;
    end
end
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
end
disp('The calculated weight matrix');
disp(w);
if x(1,1:4)==y(1:4) | x(2,1:4)==y(1:4)
    disp('The vector is a Known Vector');
else
    disp('The vector is a unknown vector');
end
```

Output:

The calculated weight matrix

```
2  2  0  0
2  2  0  0
0  0  2  2
0  0  2  2
```

The vector is an unknown vector.

Exercise 7: Consider a vector (1,0,1,1) to be stored in the net. Test a discrete Hopfield net with error in the 1st and 4th components (0,0,1,0) of the stored vector.

Description:

A Hopfield network is a form of recurrent artificial neural network popularized by John Hopfield in 1982, but described earlier by Little in 1974. Hopfield nets serve as content-addressable memory systems with binary threshold nodes. They are guaranteed to converge to a local minimum, but will sometimes converge to a false pattern (wrong local minimum) rather than the stored pattern (expected local minimum). Hopfield networks also provide a model for understanding human memory.

Aim: Write a MATLAB program to test a discrete Hopfield net with error in the 1st and 4th components (0,0,1,0) of the stored vector. Consider a vector (1,0,1,1) to be stored in the net.

Program:

```
%Discrete Hopfield net
clc;
clear;
x=[1 1 1 0];
tx=[0 0 1 0];
w=(2*x'-1)*(2*x-1);
for i=1:4
    w(i,i)=0;
end
con=1;
y=[0 0 1 0];
while con
    up=[4 2 1 3];
    for i=1:4
        yin(up(i))=tx(up(i))+y*w(1:4,up(i));
        if yin(up(i))>0
            y(up(i))=1;
        end
    end
end
if y==x
    disp('Convergence has been obtained');
    disp('The Converged Output');
    disp(y);
end
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

end
end

Output

Convergence has been obtained

The Converged Output

1 1 1 0

Exercise 8: XOR function (binaryinputandoutput) with momentumfactorusingback-propagation algorithm.

Description:

The backward propagation of errors or backpropagation, is a common method of training artificial neural networks and used in conjunction with an optimization method such as gradient descent. The algorithm repeats a two phase cycle, propagation and weight update. When an input vector is presented to the network, it is propagated forward through the network, layer by layer, until it reaches the output layer. The output of the network is then compared to the desired output, using a loss function, and an error value is calculated for each of the neurons in the output layer. The error values are then propagated backwards, starting from the output, until each neuron has an associated error value which roughly represents its contribution to the original output.

Backpropagation uses these error values to calculate the gradient of the loss function with respect to the weights in the network. In the second phase, this gradient is fed to the optimization method, which in turn uses it to update the weights, in an attempt to minimize the loss function.

The importance of this process is that, as the network is trained, the neurons in the intermediate layers organize themselves in such a way that the different neurons learn to recognize different characteristics of the total input space. After training, when an arbitrary input pattern is present which contains noise or is incomplete, neurons in the hidden layer of the network will respond with an active output if the new input contains a pattern that resembles a feature that the individual neurons have learned to recognize during their training.

Backpropagation requires a known, desired output for each input value in order to calculate the loss function gradient – it is therefore usually considered to be a supervised learning method; nonetheless, it is also used in some unsupervised networks such as auto-encoders. It is a generalization of the delta rule to multi-layered feedforward networks, made possible by using the chain rule to iteratively compute gradients for each layer. Backpropagation requires that the activation function used by the artificial neurons (or "nodes") be differentiable.

Aim: Write a Matlab program for XOR function (binaryinputandoutput) with momentum factor using back-propagation algorithm.

Program

```
%Back Propagation Network for Data Compression
clc;
clear;
%Get Input Pattern from file
data=open('comp.mat');
x=data.x;
t=data.t;
%Input,Hidden and Output layer definition
n=63;
m=63;
h=24;
%Initialize weights and bias
v=rand(n,h)—0.5;
v1=zeros(n,h);
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
b2=rand(1,m)—0.5;
w=rand(h,m)—0.5;
w1=zeros(h,m);
alpha=0.4;
mf=0.3;
con=1;
epoch=0;
while con
    e=0;
    for I=1:10
        %Feed forward
        for j=1:h
            zin(j)=b1(j);
            for i=1:n
                zin(j)=zin(j)+x(I,i)*v(i,j);
            end
            z(j)=bipsig(zin(j));
        end
        for k=1:m
            yin(k)=b2(k);
            for j=1:h
                yin(k)=yin(k)+z(j)*w(j,k);
            end
            y(k)=bipsig(yin(k));
            ty(I,k)=y(k);
        end
        %Backpropagation of Error
        for k=1:m
            delk(k)=(t(I,k)-y(k))*bipsig1(yin(k));
        end
        for j=1:h
            for k=1:m
                delw(j,k)=alpha*delk(k)*z(j)+mf*(w(j,k)—w1(j,k));
                delinj(j)=delk(k)*w(j,k);
            end
        end
        delb2=alpha*delk;
        for j=1:h
            delj(j)=delinj(j)*bipsig1(zin(j));
        end
        for j=1:h
            for i=1:n
                delv(i,j)=alpha*delj(j)*x(I,i)+mf*(v(i,j)—v1(i,j));
            end
        end
        delb1=alpha*delj;
        w1=w;
        v1=v;
        %Weight updation
        w=w+delw;
        b2=b2+delb2;
        v=v+delv;
        b1=b1+delb1;
        for k=1:k
            e=e+(t(I,k)—y(k))^2;
        end
    end
end
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
if e<0.005
    con=0;
end
epoch=epoch+1;
if epoch==30
    con=0;
end
xl(epoch)=epoch;
yl(epoch)=e;
end
disp('Total Epoch Performed');
disp(epoch);
disp('Error');
disp(e);
figure(1);
k=1;
for i=1:2
    for j=1:5
        charplot(x(k,:),10+(j-1)*15,30-(i-1)*15,9,7);
        k=k+1;
    end
end
title('Input Pattern for Compression');
axis([0 90 0 40]);
figure(2);
plot(xl,yl);
xlabel('Epoch Number');
ylabel('Error');
title('Conversion of Net');
%Output of Net after training
for I=1:10
    for j=1:h
        zin(j)=b1(j);
        for i=1:n
            zin(j)=zin(j)+x(I,i)*v(i,j);
        end
        z(j)=bipsig(zin(j));
    end
    for k=1:m
        yin(k)=b2(k);
        for j=1:h
            yin(k)=yin(k)+z(j)*w(j,k);
        end
        y(k)=bipsig(yin(k));
        ty(I,k)=y(k);
    end
end
for i=1:10
    for j=1:63
        if ty(i,j)>=0.8
            tx(i,j)=1;
        else if ty(i,j)<=-0.8
            tx(i,j)=-1;
        else
            tx(i,j)=0;
        end
    end
end
```

```
end
end
figure(3);
k=1;
for i=1:2
    for j=1:5
        charplot(tx(k,:),10+(j-1)*15,30-(i-1)*15,9,7);
        k=k+1;
    end
end
axis([0 90 0 40]);
title('Decompressed Pattern');
subfunction used:
%Plot character
function charplot(x,xs,ys,row,col)
k=1;
for i=1:row
    for j=1:col
        xl(i,j)=x(k);
        k=k+1;
    end
end
for i=1:row
    for j=1:col
        if xl(i,j)==1
            plot(j+xs-1,ys-i+1,'k*');
            hold on
        else
            plot(j+xs-1,ys-i+1,'r');
            hold on
        end
    end
end
end
function y=bipsig(x)
y=2/(1+exp(-x))-1;
function y=bipsig1(x)
y=1/2*(1-bipsig(x))*(1+bipsig(x));
```

Output

(i) Learning Rate:0.5
Momentum Factor:0.5
Total Epoch Performed
30
Error
68.8133

Exercise 9: KohonenSelfOrganizingFeature mapsin 1-Dimensionalview.

Description:

A self-organizing map (SOM) or self-organising feature map (SOFM) is a type of artificial neural network (ANN) that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a map, and is therefore a method to do dimensionality reduction. Self-organizing maps differ from other artificial neural networks as they apply competitive learning as opposed to error-correction learning (such as backpropagation with gradient descent), and in the sense that they use a neighbourhood function to preserve the topological properties of the input space. This makes SOMs useful

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

The artificial neural network introduced by the Finnish professor Teuvo Kohonen in the 1980s is sometimes called a Kohonen map or network. The Kohonen net is a computationally convenient abstraction building on work on biologically neural models from the 1970s and morphogenesis models dating back to Alan Turing in the 1950s. Like most artificial neural networks, SOMs operate in two modes: training and mapping. "Training" builds the map using input examples (a competitive process, also called vector quantization), while "mapping" automatically classifies a new input vector. A self-organizing map consists of components called nodes or neurons. Associated with each node are weight vectors of the same dimension as the input data vectors, and a position in the map space. The usual arrangement of nodes is a two-dimensional regular spacing in a hexagonal or rectangular grid. The self-organizing map describes a mapping from a higher-dimensional input space to a lower-dimensional map space. The procedure for placing a vector from data space onto the map is to find the node with the closest (smallest distance metric) weight vector to the data space vector. While it is typical to consider this type of network structure as related to feedforward networks where the nodes are visualized as being attached, this type of architecture is fundamentally different in arrangement and motivation. Useful extensions include using toroidal grids where opposite edges are connected and using large numbers of nodes. It has been shown that while self-organizing maps with a small number of nodes behave in a way that is similar to K-means, larger self-organizing maps rearrange data in a way that is fundamentally topological in character. It is also common to use the U-Matrix. The U-Matrix value of a particular node is the average distance between the node's weight vector and that of its closest neighbours. In a square grid, for instance, we might consider the closest 4 or 8 nodes (the Von Neumann and Moore neighbourhoods, respectively), or six nodes in a hexagonal grid. Large SOMs display emergent properties. In maps consisting of thousands of nodes, it is possible to perform cluster operations on the map itself.

Aim: Write a Matlab program for drawing feature maps (Kohonen Self Organizing Feature maps) in 1-Dimensional view.

Program

```
% Demonstration of Self Organizing Feature Maps using Kohonen's Algorithm
clear;
clc;
czy = input('initialisation? Y/N [Y]: ','s');
    if isempty(czy), czy = 'y' ; end
if (czy == 'y') | (czy == 'Y'),
clear
% Generation of the input training patterns. First, the form of the input domain is selected:
indom = menu('Select the form of the input domain:',...
    'a rectangle', ...
    'a triangle', ...'
    'a circle', ...
    'a ring' , ...
    'a cross',...
    'a letter A');
    if isempty(indom), indom = 2; end
% Next, the dimensionality of the output space, l, is selected.
% The output units ("neurons") can be arranged in a linear, i.e. 1-Dimensional way, or in a rectangle,
i.e., in a 2-D space.
el = menu('Select the dimensionality of the output domain:',...
    '1-dimensional output domain', ...
    '2-dimensional output domain');
    if isempty(el), el = 1; end
m1 = 12 ; m2 = 18; % m1 by m2 array of output units
if (el == 1), m1 = m1*m2 ; m2 = 1 ; end
m = m1*m2 ;
fprintf('The output lattice is %d by %d\n', m1, m2)
mOK = input('would you like to change it? Y/N [N]: ','s');
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
if (mOK == 'y') | (mOK == 'Y')
    m = 1 ;
    while ~((m1 > 1) & (m > 1) & (m < 4000))
        m1 = input('size of the output lattice: \n m1 = ');
        if (el == 2)
            m2 = input('m2 = ');
            end
            m = m1*m2 ;
        end
    end
    fprintf('The output lattice is %d by %d\n', m1, m2)
    % The position matrix V
    if el == 1
        V = (1:m1)';
    else
        [v1 v2] = meshgrid(1:m1, 1:m2); V = [v1(:) v2(:)];
    end
    % Creating input patterns
    N = 20*m ; % N is the number of input vectors
    X = rand(1, N)+j*rand(1, N) ;
    ix = 1:N;
    if (indom == 2),
        ix = find((imag(X)<=2*real(X))&(imag(X)<=2-2*real(X))) ;
    elseif (indom == 3),
        ix = find(abs(X-.5*(1+j))<= 0.5) ;
    elseif (indom == 4),
        ix = find((abs(X-.5*(1+j))<= 0.5) & (abs(X-.5*(1+j)) >= 0.3)) ;
    elseif (indom == 5),
        ix = find((imag(X)<(2/3)&imag(X)>(1/3)) | ...
            (real(X)<(2/3)&real(X)>(1/3))) ;
    elseif (indom == 6),
        ix = find((2.5*real(X)-imag(X)>0 & 2.5*real(X)-imag(X)<0.5) | ...
            (2.5*real(X)+imag(X)>2 & 2.5*real(X)+imag(X)<2.5) | ...
            (real(X)>0.2 & real(X)<0.8 & imag(X)>0.2 & imag(X)<0.4) );
    end
    X = X(ix); N = length(X);
    figure(1)
    clf reset, hold off, % resetting workspace
    plot(X, '.'), title('Input Distribution')
    % Initialisation of weights:
    W = X(1:m)'; X = X((m+1):N) ; N = N-m ;
    % as a check, the count of wins for each output unit is calculated in the matrix "hits".
    hits = zeros(m,1);
    % An Initial Feature Map
    % Initial values of the training gain, eta, and the spread, sigma of the neighborhood function
    eta = 0.4 ; % training gain
    sg2i = ((m1-1)^2+(m2-1)^2)/4 ; % sg2 = 2 sigma^2
    sg2 = sg2i ;
    figure(2)
    clf reset
    plot([0 1],[0 1],'.'), grid, hold on,
    if el == 1
        plot(W, 'b'),
    else
        FM = full(sparse(V(:,1), V(:,2), W)) ;
        plot(FM, 'b'), plot(FM, 'r') ;
    end
end
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
title(['eta = ', num2str(eta,2), ...
      ' sigma^2 = ', num2str(sg2/2,3)])
hold off,
% end of initialisation
else % continuation
eta = input('input the value of eta [0.4]: ');
if isempty(eta), eta = 0.4; end
sg2 = input(['input the value of 2sigma^2 [' , ...
            num2str(sg2i), ']: ']);
if isempty(sg2), sg2 = sg2i; end
end
reta = (0.2)^(2/N); rsigma = (1/sg2)^(2/N);
% main loop
frm = 1;
for n = 1:N
    % for each input pattern, X(n), and for each output unit which store the weight vector W(v1, v2),
    the distance between
    % X(n) and W is calculate WX = X(n) - W ; Coordinates of the winning neuron, V(kn, :), i.e., the
    neuron for which
    % abs(WX) attains minimum
    [mnm kn] = min(abs(WX)); vkn = V(kn, :);
    hits(kn) = hits(kn)+1; % utilization of neurons
    % The neighborhood function, NB, of the "bell" shape, is centered around the winning unit V(kn, :)
    rho2 = sum(((vkn(ones(m, 1), :) - V).^2), 2);
    NB = exp(-rho2/sg2);
    % Finally, the weights are updated according to the Kohonen learning law:
    W = W + eta*NB.*WX;
    % Values of "eta", and "sigma" are reduced
    if (n<N/2), %ordering and convergence phase
        sg2 = sg2*rsigma;
    else
        eta = eta*reta;
    end
    % Every 100 updates, the feature map is plotted
    if rem(n, 10) == 0
        plot([0 1],[0 1],'.'), grid, hold on,
        if el == 1
            plot(W, 'b'), plot(W, 'r'),
            else
            FM = full(sparse(V(:,1), V(:,2), W));
            plot(FM, 'b'), plot(FM, 'r');
            end
        title(['eta = ', num2str(eta,2), ...
              ' sigma^2 = ', num2str(sg2/2,3), ...
              ' n = ', num2str(n)])
        hold off,
    end
    if sum(n==round([1, N/4, N/2, 3*N/4 N]))==1
        print('-dep2', '-f2', ['Jsom2Dt', num2str(frm)])
        frm = frm+1;
    end
end
% Final presentation of the result
plot([0 1],[0 1],'.'), grid, hold on,
if el == 1
    plot(W, 'b'), plot(W, 'r').
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
FM = full(sparse(V(:,1), V(:,2), W)) ;  
plot(FM, 'b'), plot(FM.', 'r') ;  
end  
title('A Feature Map'), hold off
```

Exercise 10: Use Kohonen Self Organizing feature map to Cluster the vectors (assume four binary vectors) using own initial weights (to be assumed) and learning rate (to be assumed).

Description:

Learning Vector Quantization can be understood as a special case of an artificial neural network, more precisely, it applies a winner-take-all Hebbian learning-based approach. It is a precursor to self-organizing maps (SOM) and related to neural gas, and to the k-Nearest Neighbor algorithm (k-NN). LVQ was invented by Teuvo Kohonen. An LVQ system is represented by prototypes $W = (\omega(i), \dots, \omega(n))$ which are defined in the feature space of observed data. In winner-take-all training algorithms one determines, for each data point, the prototype which is closest to the input according to a given distance measure. The position of this so-called winner prototype is then adapted, i.e. the winner is moved closer if it correctly classifies the data point or moved away if it classifies the data point incorrectly.

Aim: Write a Matlab program to use Kohonen Self Organizing feature map to Cluster the vectors (assume four binary vectors) using own initial weights (to be assumed) and learning rate (to be assumed).

Program

```
% Learning Vector Quantization  
clc;  
clear;  
s=[1 1 0 0;0 0 0 1;0 0 1 1;1 0 0 0;0 1 1 0];  
st=[1 2 2 1 2];  
alpha=0.6;  
% initial weight matrix first two vectors of input patterns  
w=[s(1,:);s(2,:)]';  
disp('Initial weight matrix');  
disp(w);  
% set remaining as input vector  
x=[s(3,:);s(4,:);s(5,:)]';  
t=[st(3);st(4);st(5)];  
con=1;  
epoch=0;  
while con  
    for i=1:3  
        for j=1:2  
            D(j)=0;  
            for k=1:4  
                D(j)=D(j)+(w(k,j)-x(i,k))^2;  
            end  
        end  
        for j=1:2  
            if D(j)==min(D)  
                J=j;  
            end  
        end  
        if J==t(i)  
            w(:,J)=w(:,J)+alpha*(x(i,:)'-w(:,J));  
        else
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

```
end
end
alpha=0.5*alpha;
epoch=epoch+1;
if epoch==100
    con=0;
end
end
disp('Weight Matrix after 100 epochs');
disp(w);
```

Output

Initial weight matrix

```
1  0
1  0
0  0
0  1
```

Weight Matrix after **100** epochs

```
1.0000  0
0.2040  0.5615
0        0.9584
0        0.4385
```

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Course Description

Title of Course: Seminar

Course Code: MCSE181

L-T-P scheme: 0-2-0

Course Credit: 1

The overall aim of the seminar series is to help develop an emerging field at the intersection of multi-disciplinary understandings of culture and education. It will build on the existing body of work on education and culture, but its aim is explore and develop new perspectives in this area.

The objectives of the six exploratory seminars are:

- **to explore new research from a range of academic disciplines which sheds light on the questions outlined above**
- **to showcase cutting edge research on education and culture from outstanding academic researchers from the UK and internationally**
- **to bring together seminar participants from different disciplines such as Sociology, Philosophy, Psychology, Human Geography, Media Studies as well as Education and Cultural Studies**
- **to encourage and financially support the participation of PhD students**
- **to actively involve practitioners and users from each venue**
- **to engage a core group of policy makers**
- **to use the seminars to develop links between academics and stakeholders in the arts, library, media, community and educational sectors**

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Advanced DBMS
Year: 1st Year Semester: 2nd

Subject Code-MCSE201

Module Number	Topics	Number of Lectures
1	OVERVIEW OF STORAGE AND INDEXING, DISKS AND FILES	6L
	1. Data on external storage; File Organizations and indexing;	2
	2. Index data structures; Comparison of file organizations; Indexes and	2
	3. Performance tuning. Memory hierarchy; RAID; Disk space management;	1
	4. Buffer manager; Files of records; Page formats and record formats.	1
2	TREE STRUCTURED INDEXING	4L
	1. Intuition for tree indexes;	1
	2. Indexed sequential access method;	1
	3. B+ trees, Search, Insert, Delete, Duplicates,	1
	4. B+ trees in practice.	1
3	HASH-BASED INDEXING	4L
	1. Static hashing;	2
	2. Extendible hashing, linear hashing, comparisons.	2
4	OVERVIEW OF QUERY EVALUATION, EXTERNAL SORTING:	8L
	1. The system catalog; Introduction to operator evaluation; Algorithms for relational operations;	2
	2. Introduction to query optimization; Alternative plans: A motivating example;	2
	3. What a typical optimizer does. When does a DBMS sort data? A simple two-way merge sort;	2
	4. External merge sort	2
5	EVALUATING RELATIONAL OPERATORS	6L
	1. The Selection operation; General selection conditions; The Projection operation;	2
	2. The Join operation; The Set operations; Aggregate operations;	1
	3. The impact of buffering Concurrency control and recovery system: Lock based protocol, dead lock handling, time stamp based and validation based protocol, failure classification, storage, recovery algorithm,	2
	4. Recovery and atomicity, backup.	1

	A TYPICAL RELATIONAL QUERY OPTIMIZER:	4L
6	1. Translating SQL queries in to Relational Algebra;	1
	2. Estimating the cost of a plan; Relational algebra equivalences; Enumeration of alternative plans; Nested subqueries;	2
	3. Other approaches to query optimization.	1
	PHYSICAL DATABASE DESIGN AND TUNING	6L
7	1. Introduction; Guidelines for index selection, examples;	1
	2. Clustering and indexing; Indexes that enable index-only plans; Tools to assist in index selection;	2
	3. Overview of database tuning; Choices in tuning the conceptual schema;	2
	4. Choices in tuning queries and views; Impact of concurrency; DBMS benchmarking.	1
	MORE RECENT APPLICATIONS	3L
8	1. Mobile databases; Multimedia databases;	1
	2. Geographical Information Systems; Genome data management.	2
Total Number Of Hours = 41		

Faculty In-Charge

HOD, CSE Dept.

Assignment:

Module-I:

1. How do you organize file in a database?
2. What do you mean by RAID? What is indexing?

Module-II:

1. What do you mean by B+ trees
2. How do you Search, Insert, and Delete elements from B+ tree?

Module-III:

1. What do you mean by hashing? Explain Static hashing.
2. Differentiate Extendible hashing, linear hashing.

Module-IV:

1. How query optimization is done on database? Explain with proper example.
2. What a typical optimizer does in DBMS? When does a DBMS sort data? How two-way merge sort has done in DATABASE?

Module-V:

1. What is Lock based protocol? How dead lock is handled in DBMS?
2. Explain time stamp based and validation based protocol

Module-V:

1. How translation of SQL queries in to Relational Algebra is done? Explain with a example
2. What do you mean by Relational algebra equivalences?

Module-V:

1. Explain Clustering and indexing.
2. What do you mean by view and DBMS benchmarking.

Module-V:

1. What do you mean by mobile database?
2. Explain Geographical Information Systems and Genome data management.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Advanced Computer Network & Security
Year: 1stYear

Subject Code-MCSE202
Semester: Second

Module Number	Topics	Number of Lectures
Module-1	INTRODUCTION TO INTERNETWORKING:	6L
	How networks differ, how networks can be connected, connectionless internetworking, tunneling, fragmentation, overview of underlying technologies (ethernet, token ring, token bus, fddi, ppp).	
	NETWORK LAYER PROTOCOLS:	4L
	ipv4, ipv6, nat, arp, rarp, dhcp, icmp, ospf, bgp, igmp, cidr.	
	TRANSPORT LAYER PROTOCOLS:	4L
	udp, remote procedure call, rtp, tcp, tcp Tahoe, tcp Reno, tcp New Reno, tcp Sack	
Module-2	TELEPHONE SYSTEMS:	4L
	Introduction to wireless networks and cellular technology, AMPS, D-AMPS, GSM, GPRS, CDMA, Bluetooth	
	WIRELESS INTERNET:	4L
	mIPv4, mIPv6, TCP performance, I-TCP, TCP Snoop, Freeze TCP, WWP, TCP Real.	
Module-3	WIRELESS NETWORKS:	4L
	wlan: introduction, problems and solutions, protocol stack, access methods, services, WiMAX, WiFi, ZigBee. ad-hoc networks:	
	AD-HOC NETWORKS:	4L
	Introduction, routing challenges for ad-hoc networks, routing protocols (AODV, DSDV, DSR), transport protocols (ATCP, TCP-F, TCP BUS).	
Module-4	CONGESTION CONTROL:	4L
	General principles, congestion prevention policies, choke packet, RED, ECN, EFN, EFN-ACK.	
	QOS PROVISIONING	4L
	Delay guarantees, network delay, delay jitter, play out delay, admission control, QoS objectives, the RSVP approach.	
Module-5	SECURITY:	6L
	Introduction to cryptography, symmetric key and public key algorithms, Diffie-Hellman key exchange algorithm, digital signatures, IPsec, firewall, VPN, VLAN, wireless security, authentication protocols.	
Total Number Of Hours = 44		

Faculty In-Charge

HOD, CSE Dept.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Theory of Computation
Year: 1st Year

Subject Code-MCSE203
Semester: Second

Module Number	Topics	Number of Lectures
1	Models of Computation:	1L
	Models of computation-classification, properties and equivalences.	1
2	Finite Automata:	3L
	Formal definition of a Finite Automata (FA)-Examples of FA, Designing FA,DFA and NFA, regular operations	1
	Equivalence of NFA's and DFA's. FA with Epsilon-Transitions, Epsilon-Closures, Eliminating epsilon-Transitions. Applications of FAs. Mealy and Moore machine, Deadstate, Minimization of FA, Incompletely specified machine. FA on infinite inputs.	2
3	Regular Language and Grammar:	4L
	Definition of a Regular Expressions (RE), The Operators of RE – Building RE, Conversions DFA's to RE. Equivalence of RE and NFA with Epsilon-moves,- Application of RE's.	2
	Equivalence of regular grammar and FA; Properties of Regular Languages (RL), Proving Language's not to be Regular, Pumping Lemma for RL's. Applications of the Pumping Lemma. Closure Properties of RL's, Decision Properties of RL's	2
4	Context Free Language and Grammar:	5L
	Context free languages, Derivation and languages, Relationship between derivation and derivation trees, Leftmost and Rightmost Derivations. Simplification of context free grammars–Normal forms for context free grammar's, CNF and GNF.	3
	Applications of Context-Free Grammars. Non determinism vs. Ambiguity in CFLs. Closure properties of CFLs. Algorithmic properties about CFLs. Pumping Lemma for CFL.	2
5	Push Down Automata:	4L
	Definition, Acceptance by a Push Down Automata (PDA), DPDA & NPDA, example, Equivalence of PDA's and CFG's (conversion: PDA's to CFG's and reverse).	2
	Multi stack PDA. Non-determinism and power to PDAs.	2
	Turing Machine:	6L
	Unsolvable Problems. Definition, notation and Example	2

6	of Turing Machine(TM). Programming techniques- Computable languages and functions, Church Turing hypothesis.	
	Universal TM, Random Access TM. Multitape TM, Equivalence of One-Tape and Multitape TM's, Nondeterministic TM's.	2
	Conversion of RE to TM. Multi-stack PDA & TM.	2
7	Computability and Decidability:	6L
	Church-Turing Thesis, Decision Problems, Decidability and undecidability, unsolvable problems; Halting Problem of Turing Machines; Problem reduction (Turing and mapping reduction), Intractability (Hierarchy Theorems).	3
	Mapping reductions. More undecidable languages. Rice theorem. Reductions using controlled executions. RE Completeness. Reductions using computation histories.	2
	Linear Bounded Automata. Unrestricted grammars.	1
8	Computational Complexity:	6L
	Resource-constrained computation. Time Complexity- notion of complexity classes, classes P NP, NP-complete, Boolean satisfiability, NP-Completeness of CSAT and 3SAT, NP-Hard, Cooks Theorem.	2
	The concept of reduction, co- NP, polynomial Hierarchy. Some natural NP-complete problems. Space Complexity-Savich's Theorem.	2
	The class PSPACE. Optimization, search and decision problems. Approximate solutions to optimization problems.	2
9	Logic:	1
	Propositional and First-order logic and their applications to theorem proving and logic programming.	1
10	Advanced/Emerging area's:	5L
	Elementary introductions to DNA Computing.	1
	Quantum Computing	1
	Cellular Automata	1
	Circuit complexity, Structural Complexity, Parallel Complexity, Algorithmic Information.	2
Total Number Of Hours = 40		

Faculty In-Charge
CSE Dept.

HOD,

Assignment:

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Module-1 (Model of computation):

1. Define model of computation and its types.

Module-2 (Finite Automata):

1. Suppose $M = (Q, \Sigma, q_0, A, \delta)$ is an FA accepting L . We know that if $p, q \in Q$ and $p \equiv q$, then there is a string z such that exactly one of the two states $\delta^*(p, z)$ and $\delta^*(q, z)$ is in A . Show that there is an integer n such that for every p and q with $p \equiv q$, such a z can be found whose length is no greater than n , and say what n is.
2. Suppose $L \subseteq \{a, b\}^*$ and L has three equivalence classes. Suppose they can be described as the three sets $[a]$, $[aa]$, and $[aaa]$, and also as the three sets $[b]$, $[bb]$, and $[bbb]$. How many possibilities are there for the language L ? For each one, draw a transition diagram for an FA accepting it.

Module-3 (Regular Language and grammar):

1. Suppose w and z are strings in $\{a, b\}^*$. Find regular expressions corresponding to each of the languages defined recursively below. a. $a \in L$; for every $x \in L$, then wx and xz are elements of L . b. $a \in L$; for every $x \in L$, wx , xw , and xz are elements of L . c. $a \in L$; for every $x \in L$, wx and zx are in L .
2. The order of a regular language L is the smallest integer k for which $L^k = L^{k+1}$, if there is one, and ∞ otherwise. a. Show that the order of L is finite if and only if there is an integer k such that $L^k = L^*$, and that in this case the order of L is the smallest k such that $L^k = L^*$. b. What is the order of the regular language $\{a\} \cup \{aa\} \cup \{aaa\}^*$? c. What is the order of the regular language $\{a\} \cup \{aa\} \cup \{aaa\}^*$? d. What is the order of the language corresponding to the regular expression $(a^+b^*a)(b^+ab^*ab^*a)^*$?

Module-4 (Context free language and grammar):

1. Consider the CFG with productions $S \rightarrow aSbScS \mid aScSbS \mid bSaScS \mid bScSaS \mid cSaSbS \mid cSbSaS$. Does this generate the language $\{x \in \{a, b, c\}^* \mid na(x) = nb(x) = nc(x)\}$? Prove your answer.
2. Suppose that $G_1 = (V_1, \{a, b\}, S_1, P_1)$ and $G_2 = (V_2, \{a, b\}, S_2, P_2)$ are CFGs and that $V_1 \cap V_2 = \emptyset$. a. It is easy to see that no matter what G_1 and G_2 are, the CFG $G_u = (V_u, \{a, b\}, S_u, P_u)$ defined by $V_u = V_1 \cup V_2$, $S_u = S_1$, and $P_u = P_1 \cup P_2 \cup \{S_1 \rightarrow S_2\}$ generates every string in $L(G_1) \cup L(G_2)$. Find grammars G_1 and G_2 (you can use $V_1 = \{S_1\}$ and $V_2 = \{S_2\}$) and a string $x \in L(G_u)$ such that $x \notin L(G_1) \cup L(G_2)$. b. As in part (a), the CFG $G_c = (V_c, \{a, b\}, S_c, P_c)$ defined by $V_c = V_1 \cup V_2$, $S_c = S_1$, and $P_c = P_1 \cup P_2 \cup \{S_1 \rightarrow S_1S_2\}$ generates every string in $L(G_1)L(G_2)$. Find grammars G_1 and G_2 (again with $V_1 = \{S_1\}$ and $V_2 = \{S_2\}$) and a string $x \in L(G_c)$ such that $x \notin L(G_1)L(G_2)$. c. The CFG $G^* = (V, \{a, b\}, S, P)$ defined by $V = V_1$, $S = S_1$, and $P = P_1 \cup \{S_1 \rightarrow S_1S_1 \mid \}$ generates every string in $L(G_1)^*$. Find a grammar G_1 with $V_1 = \{S_1\}$ and a string $x \in L(G^*)$ such that $x \notin L(G_1)^*$.
3. In each case below, show that the grammar is ambiguous, and find an equivalent unambiguous grammar. a. $S \rightarrow SS \mid a \mid b$; $S \rightarrow ABA \mid A \rightarrow aA \mid \text{null}; B \rightarrow bB \mid \text{null}; S \rightarrow aSb \mid aaSb \mid \text{null}; S \rightarrow aSb \mid abS \mid \text{null}$.

Module-5 (Push down automata):

1. Give transition tables for PDAs accepting each of the following languages. a. The language of all odd-length strings over $\{a, b\}$ with middle symbol a . b. $\{anx \mid n \geq 0, x \in \{a, b\}^* \text{ and } |x| \leq n\}$. c. $\{aibjck \mid i, j, k \geq 0 \text{ and } j = i \text{ or } j = k\}$.
2. Suppose $L \subseteq \text{_____}^*$

is accepted by a PDA M , and for some fixed k , and every $x \in \Sigma^*$, at least one choice of moves allows M to process x completely so that the stack never contains more than k elements. Does it follow that L is regular? Prove your answer.

Module-6 (Turing machine):

1. Let $T = (Q, \Sigma, q_0, \delta)$ be a TM, and let s and t be the sizes of the sets Q and Σ , respectively. How many distinct configurations of T could there possibly be in which all tape squares past square n are blank and T 's tape head is on or to the left of square n ? (The tape squares are numbered beginning with 0.)
2. Given TMs $T_1 = (Q_1, \Sigma, q_1, \delta_1)$ and $T_2 = (Q_2, \Sigma, q_2, \delta_2)$, with $1 \subseteq 2$, give a precise definition of the TM $T_1T_2 = (Q, \Sigma, q_0, \delta)$. Say precisely what Q, Σ, q_0 , and δ are.

Module-7 (Computability and Decidability):

1. Let $f: \mathbb{N} \rightarrow \mathbb{N}$ be the function defined as follows: $f(0) = 0$, and for $n > 0$, $f(n)$ is the maximum number of moves a TM with n non-halting states and tape alphabet $\{0, 1\}$ can make if it starts with input 1^n and eventually halts. Show that f is not computable.
2. Suppose we define $b_2(0)$ to be 0, and for $n > 0$ we define $b_2(n)$ to be the largest number of 1's that can be left on the tape of a TM with two states and tape alphabet $\{0, 1\}$, if it starts with input 1^n and eventually halts. a. Give a convincing argument that b_2 is computable. b. Is the function b_k (identical to b_2 except that "two states" is replaced by "k states") computable for every $k \geq 2$? Why or why not?

Module-8 (Computational Complexity):

1. Let L_1 and L_2 be languages over Σ_1 and Σ_2 , respectively. Show that if $L_1 \leq_p L_2$, then $L_1 \leq_p L_2$.
2. Show that if there is an NP-complete language L whose complement is in NP, then the complement of every language in NP is in NP.
3. Suppose $L_1, L_2 \in \Sigma^*$ can be accepted by TMs with time complexity τ_1 and τ_2 , respectively. Find appropriate functions g and h such that $L_1 \cup L_2$ and $L_1 \cap L_2$ can be accepted by TMs with time complexity in $O(g)$ and $O(h)$, respectively.

Module-9 (Logic):

1. Prove that For every three positive integers i, j , and n , if $ij = n$, then $i \leq \sqrt{n}$ or $j \leq \sqrt{n}$.
2. A principle of classical logic is modus ponens, which asserts that the proposition $(p \rightarrow q) \rightarrow q$ is a tautology, or that $p \rightarrow (p \rightarrow q)$ logically implies q . Is there any way to define the conditional statement $p \rightarrow q$, other than the way we defined it, that makes it false when p is true and q is false and makes the modus ponens proposition a tautology? Explain.

Module-10(Advanced/Emerging areas):

3. Write a short note on cellular automata, quantum computing.
4. Define Circuit complexity, Structural Complexity, Parallel Complexity.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Cluster, Grid, and Cloud computing
Year: 1st Year

Subject Code-MCSE204A
Semester: Second

Module Number	Topics	Number of Lectures
1	Cluster Computing:	12L
	A general introduction to the concept of cluster based distributed computing.	1
	Hardware technologies for cluster computing	1
	A survey of the possible node hardware and high-speed networking hardware and software.	1
	Software architectures for cluster computing including both shared memory (OpenMP) and message-passing (MPI/PVM) models MPI-2 extension, dynamic process creation, one-sided communication, parallel I/O	4
	Variants based on new low level protocols (MVAPICH).	2
	Evaluation and tuning of system and software performance evaluation tools.	1
	HINT, netperf, netpipe, tcp, lperf.	2
2	Grid Computing:	12L
	The Grid - Past, Present, Future, A New Infrastructure for 21st Century Science – The Evolution of the Grid - Grids and Grid Technologies.	2
	Programming models - A Look at a Grid Enabled Server and Parallelization Techniques – Grid applications.	2
	The concept of virtual organizations – Grid architecture – Grid architecture and relationship to other Distributed Technologies – computational and data Grids.	3
	Semantic grids Case Study: Molecular Modeling for Drug Design and Brain Activity Analysis.	2
	Resource management and scheduling, Setting up Grid.	2
	Deployment of Grid software and tools, and application execution.	1

	Cloud Computing:	12L
3.	Introduction to Cloud Computing, Definition, Characteristics, Components, Cloud provider,	1
	SAAS,PAAS IAAS and Others, Organizational scenarios of clouds.	2
	Administering & Monitoring cloud services, benefits and limitations,Deploy application over cloud, Comparison among SAAS, PAAS, IAAS Cloud computing platforms:	2
	Infrastructure as service: Amazon EC2, Platform as Service: Google App Engine, Microsoft Azure, Utility Computing.	2
	Elastic Computing Data in the cloud: Relational databases, Cloud file systems: GFS and HDFS, BigTable, HBase and Dynamo.	2
	Issues in cloud computing, Implementing real time application over cloud platform Issues in Inter cloud environments.	2
	QOS Issues in Cloud, Dependability, data migration, streaming in Cloud. Quality of Service (QoS) monitoring in a Cloud computing environment.	1
Total Number Of Hours = 42		

Faculty In-Charge

HOD, CSE Dept.

Assignment:

Module-1:

1. Define Cluster computing? Why we need the cluster computing?
2. Describe system architecture for cluster computing.
3. How cluster can be formed?

Module-2:

1. How cluster computing differs from grid computing?
2. Describe the grid architecture
3. How grid computing involves in Molecular Modelling for Drug Design and Brain Activity Analysis?

Module-3:

1. Describe all the cloud services?
2. Is cloud a service? Justify your answer.
3. Discuss the security issues of cloud.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Mobile Computing

Subject Code-MCSE204B

Year: 1stYear

Semester: 2nd

Module Number	Topics	Number of Lectures
1	Fundamentals of Cellular Communications	8L
	1. Introduction, First- and Second-Generation Cellular Systems, Cellular Communications from 1G to 3G,	2L
	2. Tele-traffic Engineering, Radio Propagation and Propagation Path-Loss Models, Cellular Geometry, Interference in Cellular Systems,	2L
	3. Frequency Management and Channel Assignment Issues, Multiple Access Techniques,	2L
	4. GSM Logical Channels and Frame Structure, Privacy and Security in GSM, Mobility Management in Cellular Networks.	2L
2	Wireless Transmission Fundamentals	8L
	1. Spread Spectrum (SS) and CDMA Systems, Wireless Medium Access Control, IEEE 802.11 Architecture and Protocols,	2L
	2. Issues in Ad Hoc Wireless Networks (Medium Access Scheme), Routing, Multicasting, Transport Layer Protocols, QoS Provisioning,	2L
	3. Energy Management and Energy Consumption Models, Traffic Integration in Personal, Local, and Geographical Wireless Networks,	2L
	4. Bluetooth, Technologies for High-Speed WLANs, Third-Generation Cellular Systems: UMTS.	2L
3	Mobile Adhoc Networks	10L
	1. Introductory Concepts. Different models of operation, various applications of MANET,	1L
	2. Destination-Sequenced Distance Vector protocol - overview, Route Advertisement, Extending Base Station Coverage, Properties of DSDV protocol,	2L
	3. Dynamic Source Routing protocol - overview and properties, DSR Route Discovery, Route Maintenance, Support for Heterogeneous Networks and Mobile IP, Multicast routing with DSR,	2L
	4. Ad Hoc On-Demand Distance-Vector protocol - properties, Unicast Route Establishment, Multicast Route Establishment, Broadcast Optimizations and Enhancements,	2L
	5. Link Reversal Routing - Gafni-Bertsekas Algorithm, lightweight mobile routing algorithm, Temporally Ordered Routing Algorithm,	2L
	6. Preserving battery life of mobile nodes - Associativity Based Routing, Effects of	1L

	beaconing on battery life.	
4	Wireless Sensor Networks	9L
	1. Sensor networks overview: introduction, applications, design issues, requirements, Sensor node architecture, Network architecture: optimization goals, evaluation metrics, network design principles,	2L
	2. Sensor network operating systems and brief introduction to sensor network Programming, Network protocols: MAC protocols and energy efficiency,	2L
	3. Routing protocols: data centric, hierarchical, location-based, energy efficient routing etc, Sensor deployment, scheduling and coverage issues,	2L
	4. Self-Configuration and Topology Control, Querying, data collection and processing, collaborative information processing and group connectivity,	2L
	5. Target tracking, localization and identity management, Power management, Security and privacy.	1L
5	Topology Control and Clustering in Adhoc Networks	5L
	1. Algorithms for Graphs Modeling Wireless Ad Hoc Networks,	2L
	2. Clustering and Network Backbone, Dominating Set Based Routing in Ad Hoc Wireless Networks,	2L
	3. Formation of a Connected Dominating Set, Backbone-Formation Heuristics.	1L
6	Mobile, Distributed and Pervasive Computing	5L
	1. Pervasive Computing Applications,	1L
	2. Architecture of Pervasive Computing Software,	2L
	3. Indoor Wireless Environments, Challenges for the Future: Nomadic Computing.	2L
Total Number Of Lectures = 45		

Faculty In-Charge

HOD, CSE Dept.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Mobile Computing
Year: 1stYear

Semester: 2nd

Subject Code-MCSE204B

Assignment:

Module-1(Fundamentals of Cellular Communications):

1. What is digital modulation? What is FSK, PSK, and QAM? What is 'handover' in mobile system? How it is effected in GSM?
2. A. Define BTS. What is a MSC? What are the functions of MSC in network and switching subsystem?
B. A certain city has an area of 1300 square miles and is covered by a cellular system using a seven cell reuse pattern. Each cell has a radius of 4 miles and the city has 40 MHz spectrum with a full duplex channel bandwidth of 60KHz. Find:
(i) The number of cells in the service area.
(ii) The number of channels per cell.
(iii) Total number of subscribers that can be served.

Module-2 (Wireless Transmission Fundamentals):

1. State the modes possible when the slave is in connection state in Bluetooth. Explain the concepts of CDMA. What are its merits and demerits? Explain the working principle of RAKE receiver.
2. Explain the TDMA frame structure and derive the efficiency of a TDMA system. What are the different types of Hand over?

Module-3(Mobile Adhoc Networks):

1. What is the necessity of MANET in mobile computing? What do you mean by Mobile Ad hoc Network? Explain DSDV protocol. How route advertisement is occurs in DSDV? Explain the power consumption mechanism in Ad hoc network.
2. Explain AODV protocol. How unicast route establishment, multicast route establishment happens in AODV. Explain Link Reversal Routing algorithm with proper diagram.

Module-4(Wireless Sensor Networks):

1. What do you mean by node in WSN? Write down its architecture. How you manage the security of a WSN node?
2. Explain MAC in WSN. How power of a node should be managed in WSN? How self-configuration should be done in WSN node? How data packets are transmitted through WSN nodes? How connection is done in all the nodes of WSN?

Module-5(Topology Control and Clustering in Adhoc Networks):

1. What do you mean by backbone of a network? How clustering is happens in Ad hoc nodes? How you can manage heuristic backbone node formation?
2. What do you mean by Dominating Set Based Routing in Ad Hoc Wireless Networks? What do you mean by clustering? By using which approach clustering is done in WSN?

Module-6 (Mobile, Distributed and Pervasive Computing):

1. What do you mean by Pervasive Computing? Explain its applications.
2. Explain the Architecture of Pervasive Computing Software. How you can set the indoor wireless environment?

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Advanced Web Technology

Subject Code-MCSE204C

Year: 1st Year

Semester: Second

Module Number	Topics	Number of Lectures
1	Introduction:	6L
	1. Overview, Computer Network, Intranet, Extranet and Internet. Types of Networks (LAN, MAN, WAN), Network Topologies .Definition of Internet, Internet organization. Growth of Internet, Internet Application.	2
	2. Review of TCP/IP: OSI Reference model, TCP/IP Model, IP addressing, Classful and Classless Addressing, Subnetting, Features and services of TCP/IP, Three-Way Handshaking, Flow Control, Error Control, Congestion control, IP Datagram. Routing - Intra and Inter Domain Routing, Unicast and Multicast Routing, Broadcast. Electronic Mail-POP3, SMTP.	2
	3. World Wide Web: Evolution of distributed computing. Core distributed computing technologies – Client/Server Architecture & its Characteristics, JAVA RMI. Challenges in Distributed Computing, role of J2EE and XML in distributed computing, emergence of Web Services and Service Oriented Architecture (SOA). Introduction to Web Services – The definition of web services, basic operational model of web services, tools and technologies enabling web services, benefits and challenges of using web services. Web Server Concept and Architecture. Definition of DNS (Domain Name System). Domain and Sub domain, Address Resolution, FTP & its usage, Telnet Concepts, Remote Logging, HTTP & HTTPs.	2
2	Client Side Application Development	12L
	1. HTML & CSS: Introduction, Editors, Elements, Tags, Attributes, Heading, Paragraph. Formatting, Link, Image, Table, List, Block, Form, Frame Layout, DHTML, Basic Web Page Development, CSS- Create Class Styles, Create ID Styles ,Span, Colors.HTML5 in brief.	3L
	2. Extensible Markup Language (XML): Brief Over View of XML – XML Document structure, XML namespaces, Defining structure in XML documents, Reuse of XML schemes, Document navigation and transformation, Tree, Syntax, Elements, Attributes, Validation, and Viewing. XHTML in brief.	3L

	3. JavaScript: Introduction, JavaScript in Web Pages, The Advantages of JavaScript Writing JavaScript into HTML; Building Up JavaScript Syntax; Basic Programming Techniques ; Operators and Expressions in JavaScript; JavaScript Programming Constructs; Conditional Checking Functions in JavaScript, Dialog Boxes, Statements, comments, variable, comparison, condition, switch, loop, break. Object – string, array. Function, Errors, Validation. The JavaScript Document Object Model-Introduction (Instance, Hierarchy); The JavaScript Assisted Style Sheets DOM; Understanding Objects in HTML (Properties of HTML objects, Methods of HTML objects); Browser Objects, Handling Events Using JavaScript	6L
	Server Side Programming with PHP & MySQL	16L
3	1. Installing and Configuring : Current and Future Versions of MySQL and PHP, How to Get MySQL, Installing MySQL on Windows, Trouble Shooting your Installation, Basic Security Guidelines, Building PHP on Windows with Apache, Windows, php.ini.Basics,The Basics of PHP scripts.	2L
	2. The Building blocks of PHP: Variables, Data Types, Operators and Expressions, Constants. Flow Control Functions in PHP: Switching Flow, Loops, Code Blocks and Browser Output.	3L
	3. Functions: What is function? Calling functions, Defining Functions. Variable Scope, more about arguments. Working with Arrays and Some Array-Related Functions.	3L
	4. Working with Objects: Creating Objects, Object Instance Working with Strings, Dates and Time: Formatting strings with PHP, Investigating Strings with PHP, Manipulating Strings with PHP, Using Date and Time Functions in PHP.	2L
	5. Working with Forms: Creating Forms, Accessing Form Input with User defined Arrays, Combining HTML and PHP code on a single Page, Using Hidden Fields to save state, Redirecting the user, Sending Mail on Form Submission, and Working with File Uploads.	2L

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Advanced Web Technology

Subject Code-MCSE204C

Year: 1st Year

Semester: Second

	6. Learning basic SQL Commands: Learning the MySQL Data types, Learning the Table Creation Syntax, Using Insert Command, Using SELECT Command, Using WHERE in your Queries, Selecting from Multiple Tables, Using the UPDATE command to modify records, Using the DELETE Command, Frequently used string functions in MySQL, Using Date and Time Functions in MySQL.	2L
	7. Interacting with MySQL using PHP: MySQL Versus MySQLi Functions, Connecting to MySQL with PHP, Working with MySQL Data.	2L
	Multimedia for WEB	6L
4	1. Multimedia Application Development: Pixel, Image Resolution, Image Editing using Photoshop, 2D & 3D Animation, Logo Design, Banner. Animated Component Preparation using Flash & Action script.	4L
	2. Multimedia Web Applications: Multimedia over IP: RTP, RTCP. Streaming media, Codec and Plugins, VoIP, Text and Voice Chat.	2L

Faculty In-Charge

HOD, CSE Dept.

Assignment:

Module-1 (Introduction):

1. Internet Application.

Module-2 (Client Side Application Development):

1. Design a web site for student management system using HTML, CSS and JavaScript.

Module-3 (Server Side Programming with PHP & MySQL):

1. Use php and MySQL in the previous student management system to make the site complete.

Module-4 (Multimedia for WEB):

1. Use some multimedia applications in student management system.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Soft Computing
Year: 1stYear

Semester: 2nd

Subject Code-MCSE204D

Module Number	Topics	Number of Lectures
1	Introduction to Soft Computing:	8L
	1. Evolution of Computing	2
	2. Soft Computing Constituents	2
	3. From Conventional Artificial Intelligence to Computational Intelligence	2
	4. Machine Learning Basics.	2
2	Fuzzy Logic:	8L
	1. Fuzzy sets and Fuzzy logic: Introduction, Fuzzy sets versus crisp sets, operations on fuzzy sets, Extension principle	2
	2. Fuzzy relations and relation equations, Fuzzy numbers, Linguistic variables	2
	3. Fuzzy logic, Linguistic hedges, Applications, fuzzy controllers	2
	4. Fuzzy pattern recognition, fuzzy image processing, fuzzy database.	2
3	Artificial Neural Networks:	8L
	1. Artificial Neural Network: Introduction, basic models,	2
	2. Hebb's learning, Adaline, Perceptron, Multilayer feed forward network, Back propagation	2
	3. Different issues regarding convergence of Multilayer Perceptron, Competitive learning	2
	4. Self-Organizing Feature Maps, Adaptive Resonance Theory, Associative Memories, Applications.	2
4	Genetic Algorithms:	8L
	1. Evolutionary and Stochastic techniques: Genetic Algorithm (GA), different operators of Genetic Algorithm,	2
	2. Analysis of selection operations, Hypothesis of building Blocks, Schema theorem and convergence of Genetic Algorithm, Simulated annealing and Stochastic models,	3
	3. Boltzmann Machine, Applications. Rough Set: Introduction, Imprecise Categories Approximations and Rough Sets, Reduction of Knowledge, Decision Tables, and Applications.	3
5	Hybrid Systems:	8L
	1. Neural-Network-Based Fuzzy Systems,	2
	2. Fuzzy Logic-Based Neural Networks, Genetic Algorithm for Neural Network Design and Learning,	3
	3. Fuzzy Logic and Genetic Algorithm for Optimization, Applications.	3
Total Number Of Lectures = 40		

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Soft Computing
Year: 1stYear

Semester: 2nd

Subject Code-MCSE204D

Assignment:

Module-1(Introduction to Soft Computing):

1. Explain Soft Computing Constituents.
2. How evaluation is done from Conventional Artificial Intelligence to Computational Intelligence. Define Machine Learning Basics.

Module-2 (Fuzzy Logic):

1. Explain Fuzzy relations and relation equations.What do you mean by Linguistic variables?
2. Define Linguistic hedges and App lications.

Module-3(Artificial Neural Networks):

1. What do you mean by Artificial Neural Network? Explain its basic models.
2. Explain the different issues regarding convergence of Multilayer Perceptron andCompetitive learning. What is Self-Organizing Feature Maps?

Module-4(Genetic Algorithms):

1. Explain the Schema theorem and convergence of Genetic Algorithm.
2. What issimulated annealing and stochastic models and Boltzmann Machine?

Module-5(Hybrid Systems):

1. Explain Neural-Network-Based Fuzzy Systems with proper example. What is Fuzzy Logic-Based Neural Networks?
2. Explain any one Genetic Algorithm for Neural Network Design.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Cryptography & Computer Security Subject Code-MCSE204E

Year: 1st Year, M.Tech.Semester: Second

Module Number	Topics	Number of Lectures
1	Introduction:	8L
	1. Linear algebra: non linearity, echelon form of matrix, Galois Field, vector space, Modular arithmetic	3
	2. Coding Theory : Huffman coding, Hamming Coding,	2
	3. Number Theory : Prime Numbers, Modular Arithmetic and Discrete Logarithms, Fermet's Thoerem, Eulers Theorem, Chinese Remainder Theorem, Quadratic Reciprocity Theorem Information Theory : Entropy and Uncertainty	3
2.	Various Types of Attacks:	4L
	1. Attacks on Computers & Computer Security	2
	2. Need for Security, Security approaches,	1
	3. Principles of Security, Types of attack	1
3.	Cryptography: Concepts & Techniques	6L
	1. Introduction, Plaintext & Cipher text, Substitution Techniques	2
	2. Transposition Techniques, Encryption & Decryption,	2
	3. Symmetric & Asymmetric key Cryptography, Key Range & Key Size	2
4.	Symmetric Key Algorithm	6L
	1. Introduction, Algorithm types & Modes, Overview of Symmetric Key Cryptography	2
	2. DES(Data Encryption Standard) algorithm,	2
	3. IDEA(International Data Encryption Algorithm) algorithm, RC5(Rivest Cipher 5) algorithm.	2
5.	Asymmetric Key Algorithm, Digital Signature and RSA	6L
	1. Introduction, Overview of Asymmetric key Cryptography	2
	2. RSA algorithm, Symmetric & Asymmetric key Cryptography together,	2
	3. Digital Signature, Basic concepts of Message Digest and Hash Function	2
6.	Internet Security Protocols, User Authentication	6L
	1. Internet Security Protocols, User Authentication	2
	2. Authentication Basics, Password, Authentication Token	2
	3. Certificate based Authentication, Biometric Authentication	2

7.	Electronic Mail Security	4L
	1. Basics of mail security	2
	2. Pretty Good Privacy, S/MIME	2
8.	Firewall	4L
	1. Introduction, Types of firewall	2
	2. Firewall Configurations, DMZ Network	2
Total Number Of Hours = 44		

Faculty In-Charge

HOD, CSE Dept.

Assignment :

Module -1 (Introduction)

Module -2 (Various Types of Attacks)

Module -3 (Cryptography: Concepts & Techniques)

Module -4 (Symmetric Key Algorithm)

Module -5 (Asymmetric Key Algorithm, Digital Signature and RSA)

Module -6 (Internet Security Protocols, User Authentication)

Module -7 (Electronic Mail Security)

Module -8 (Firewall)

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Image Processing Subject Code-MCSE205A

Year: 1stYear, M.Tech.Semester: Second

Module Number	Topics	Number of Lectures
1.	Introduction:	4L
	1. Background, Digital Image Representation,	1
	2. Fundamental steps in Image Processing	1
	3. Elements of Digital Image Processing – Image Acquisition, Storage, Processing, Communication, Display.	2
2.	Digital Image Formation	6L
	1. A Simple Image Model, Geometric Model- Basic Transformation (Translation, Scaling, Rotation),	3
	2. Perspective Projection,	1
	3. Sampling & Quantization - Uniform & Non-uniform	2
3.	Mathematical Preliminaries	8L
	1. Neighbour of pixels, Connectivity, Relations, Equivalence & Transitive Closure;	2
	2. Distance Measures, Arithmetic/Logic Operations, Fourier Transformation,	3
	3. Properties of The Two Dimensional Fourier Transform, Discrete Fourier Transform, Discrete Cosine & Sine Transform	3
4.	Image Enhancement	10L
	1. Spatial Domain Method, Frequency Domain Method, Contrast Enhancement –Linear & Nonlinear Stretching	3
	2. Histogram Processing; Smoothing - Image Averaging, Mean Filter, Low-pass Filtering; Image Sharpening. High-pass Filtering, High-boost Filtering	4
	3. Derivative Filtering, Homomorphic Filtering; Enhancement in the frequency domain - Low pass filtering, High pass filtering.	3
5.	Image Restoration	6L
	1. Degradation Model, Discrete Formulation, Algebraic Approach to Restoration – Unconstrained & Constrained	2
	2. Constrained Least Square Restoration, Restoration by Homomorphic Filtering,	2
	3. Geometric Transformation - Spatial Transformation, Gray Level Interpolation	2
6.	Image Segmentation	8L
	1. Point Detection, Line Detection, Edge detection, Combined detection, Edge Linking & Boundary Detection - Local Processing	3
	2. Global Processing via The Hough Transform; Thresholding - Foundation, Simple Global Thresholding, Optimal Thresholding	3
	3. Region Oriented Segmentation - Basic Formulation, Region Growing by Pixel Aggregation, Region Splitting & Merging	2
Total Number Of Hours = 42		

Assignment :

Module -1 (Introduction)

Module -2 (Digital Image Formation)

Module -3 (Mathematical Preliminaries)

Module -4 (Image Enhancement)

Module -5 (Image Restoration)

Module -6 (Image Segmentation)

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: *Pattern Recognition*
Year: 1st Year

Subject Code: *MCSE205B*
Semester: **2nd**

Module Number	Topics	Number of Lectures
1	INTRODUCTION TO PATTERN RECOGNITION:	6L
	1. Basic concepts- Definitions, data sets for Pattern Recognition	1L
	2. Structure of a typical pattern recognition system.	1L
	3. Different Paradigms of Pattern Recognition.	1L
	4. Representations of Patterns and Classes.	1L
	5. Metric and non-metric proximity measures.	2L
2	FEATURES SELECTION:	5L
	1. Feature vectors - Feature spaces	2L
	2. Different approaches to Feature Selection-Branch and Bound Schemes.	2L
	3. Sequential Feature Selection.	1L
3	FEATURES EXTRACTION :	4L
	1. Principal Component Analysis (PCA)	2L
	2. Kernel PCA	2L
4	PATTERN CLASSIFICATION:	12L
	1. Pattern classification using Statistical classifiers - Bayes' classifier.	2L
	2. Classification performance measures – Risk and error probabilities.	2L
	3. Linear Discriminant Function, Mahalanobis Distance.	2L
	4. K-NN Classifier, Fisher's LDA	2L
	5. Single Layer Perceptron, Multi-layer Perceptron	2L
	6. Training set, test set; standardization and normalization	2L
5	CLUSTERING	8L

	1. Basics of Clustering; similarity / dissimilarity measures; clustering criteria.	3L
	2. Different distance functions and similarity measures.	2L
	3. K-means algorithm, K-medoids, DBSCAN	3L
	RECENT ADVANCES IN PATTERN RECOGNITION	3L
6	1. Structural PR, SVMs, FCM	1L
	2. Soft-computing and Neuro-fuzzy techniques, and real life examples.	2L
Total Number Of Hours = 38L		

Faculty In-Charge

HOD, CSE Dept.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Real-Time Embedded System and Programming Subject Code: MCSE205C

Year: 1st Year

Semester: 2nd

Module Number	Topics	Number of Lectures
1	Introduction to Embedded Systems	3L
	1. Definition of Embedded System, Embedded Systems Vs General Computing Systems, History of Embedded Systems	1
	2. Classification, Major Application Areas, Purpose of Embedded Systems	1
	3. characteristics and Quality Attributes of Embedded Systems	1
2	Typical Embedded System	9L
	1. Core of the Embedded System: General Purpose and Domain Specific Processors	1
	2. ASICs, PLDs, Commercial Off-The-Shelf Components (COTS)	2
	3. Memory: ROM, RAM, Memory according to the type of Interface, Memory Shadowing, Memory selection for Embedded Systems	3
	4. Sensors and Actuators	
	5. Communication interface : onboard and external communication interface	2 1
3	Embedded Firmware	5L
	1. Reset Circuit, Brown-out Protection Circuit	1
	2. Oscillator Unit, Real Time Clock	2
	3. Watchdog Timer	1
	4. Embedded Firmware Design Approaches and Development Languages	1
4	RTOS Based Embedded System Design	5L
	1. Operating System Basics, Types of Operating Systems	1
	2. Tasks, Process and Threads	2
	3. Multiprocessing and Multitasking	1
	4. Task Scheduling	1
5	Task Communication	7L
	1. Shared Memory, Message Passing	2
	2. Remote Procedure Call and Sockets	2
	3. Task Communication/Synchronization Issues	1
	4. Task Synchronization Techniques, Device Drivers, How to Choose an RTOS.	2
Total Lecture Hours – 29		

Faculty In-Charge

HOD, CSE Dept.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Real-Time Embedded System and Programming Subject Code: MCSE205C

Year: 1st Year

Semester: 2nd

Assignment:

Module-I: Introduction to Embedded Systems

1. Explain Embedded System. What are the differences between Embedded Systems and General Computing Systems?
2. What are the purposes of using embedded systems? Explain the Quality Attributes of Embedded Systems.

Module-II: Typical Embedded System

1. Explain Memory Shadowing and Memory selection for Embedded Systems.
2. What do you mean by Domain Specific Processors of Embedded systems?

Module-III: Embedded Firmware

1. Explain different Embedded Firmware Design Approaches and Development Languages.
2. What is the purpose of using Brown-out Protection Circuit?

Module-IV: RTOS Based Embedded System Design

1. What are threads? How Task Scheduling happens in embedded systems?
2. Explain real time operating system with a proper example.

Module-V: Task Communication

1. What do you mean by Remote Procedure Call?
2. What are the Task Synchronization issues required to design RTOS?

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Complex Systems
Year: 1st Year

Subject Code-MCSE205D
Semester: Second

Module Number	Topics	Number of Lectures
1	Introduction:	4L
	Non-linear complex systems.	2
	mathematical description in the shape of subsystems and interactions between them	2
2	Special matrices:	6L
	Special matrices	3
	Special matrices and their use for analysis of complex systems	3
3.	Eigen Value:	4L
	Estimation of eigen values of aggregation matrices	2
	Estimation of border interactions	2
4	Decomposition and Aggregation:	4L
	Principles of Decomposition	2
	Principles of Aggregation	2
5	Lyapunov method:	4L
	The direct Lyapunov method overview	1
	Problem solving and case studies on Lyapunov method	3
6	Stability analysis:	4L
	Stability analysis of complex nonlinear systems	1
	Stability analysis of complex nonlinear systems by the method of vector Lyapunov function	3
Total Number Of Hours = 26L		

Faculty In-Charge

HOD, CSE Dept.

Assignment:

Module-1:

1. Define Complex System
2. Write down the features of Complex System.

Module-2:

1. Why we need special matrices?
2. How Special matrices are defer from normal matrices.

Module-3:

1. How we estimate the eigen values of aggregation matrices

Module-4:

1. State the principle of decomposition.
2. State the principle of aggregation.

Module-5:

1. Discuss Lyapunov method in brief.

Module-6:

1. How we can analysis the stability of non linear complex system.
2. How vector Lyapunov function helps to analyze the stability of complex system.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Distributed System

Subject Code-MCSE205E

Year: 1st Year

Semester: Second

Module Number	Topics	Number of Lectures
1	Basic concepts:	13L
	Models of computation: shared memory and message passing systems, synchronous and asynchronous systems.	2
	Logical time and event ordering.	2
	Global state and snapshot algorithms, mutual exclusion, clock synchronization	2
	Leader election, deadlock detection,	4
	Termination detection, spanning tree construction.	3
2	Programming models:	3L
	Remote procedure calls,	1
	Distributed shared memory.	2
3.	Fault tolerance and recovery:	12L
	Basic concepts, fault models,	2
	Agreement problems and its applications,	4
	Commit protocols, voting protocols,	3
	Check pointing and recovery, reliable communication.	3
4	Security and Authentication:	4L
	Basic concepts, Kerberos. Resource sharing and load balancing.	2
	Resource sharing and load balancing.	2
5	Special topics:	4L
	Distributed objects, distributed databases, directory services,	2
	Directory services,	2
Total Number Of Hours = 36L		

Faculty In-Charge

HOD, CSE Dept.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Computer Networking & DBMS Laboratory Lab

Course Code: MCSE291

L-T-P Scheme: 0-0-3

Course Credits: 2

Objective:

- To get familiar with basic EER concepts (generalization and specialization)
- Study of ODL schema.
- Implement Abstract data type.
- To study varying array.
- Implement object Table.
- To study implementation of nested tables.
- To understand use of objects in member procedure.
- To study horizontal Partitioning
- To study horizontal Partitioning.
- To study semistructured data.

Learning Outcomes:

- Ability to build normalized databases.
- Knowledge of Entity Relationship Modelling.
- Familiarity with SQL, embedded SQL and PLSQL, ODL schema.
- Familiarity with query processing and query optimization techniques.
- Understanding of Member procedure., Partitioning on the tables.
- Ability to handle recovery and concurrency issues.
- Familiarity with XML commands.

Course Contents:

Exercises that must be done in this course are listed below:

Exercise No.1: Study of EER diagram.

Exercise No. 2: Study of ODL schema.

Exercise No. 3: Implement Abstract data type.

Exercise No. 4: Implement Varrays.

Exercise No. 5: Implement object Table.

Exercise No. 6: Implement Nested table.

Exercise No. 7: Implement Member procedure.

Exercise No. 8: Implement Member Function.

Exercise No. 9: Implement Partitioning on the tables.

Exercise No. 10: Study of XML commands.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Lab Assignment 1	
Title	Study of EER diagram.
Objective	To get familiar with basic EER concepts (generalization and specialization)
References	Fundamentals of Database Systems Elmasri and Navathe Addison-Wesley Fourth edition Database management Systems Raghu Ramakrishnan and Johannes Gehrke TMH Database System concepts Korth,Silberchatz,sudarshanMc-Graw-HillFourth edition Database Systems, design, Implementation and management Peter Rob and Carlos Coronel Thomson learning. Fifth edition Introduction to database systems C.J Date Addison Wesley Longman
Pre-requisite	Knowledge of →basic ER model. →Specialization and generalization.
Theory	A case study is being considered for studying EER concepts. The EER model includes all modeling concepts of ER model, in addition to that it includes concepts of subclass, superclass, specialization and generalization. Another concept included is that of a category or union type which is used to represent a collection of objects.
Sample Output	A detailed EER diagram and information related to all the entity sets.
Post Lab Assignment	Explain Generalization and specialization? Constraints on Generalization and specialization?

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Lab Assignment 2

Lab Assignment 2	
Title	Study of ODL schema.
Objective	To get familiar object structure, naming mechanism and ODL.
References	Fundamentals of Database Systems Elmasri and Navathe Addison-Wesley Fourth edition Database management Systems Raghu Ramakrishnan and Johannes Gehrke TMH Database System concepts Korth, Silberchatz, sudarshan Mc-Graw-Hill Fourth edition Database Systems, design, Implementation and management Peter Rob and Carlos Coronel Thomson learning. Fifth edition Introduction to database systems C.J Date Addison-Wesley Longman
Pre-requisite	Knowledge of → EER model. DDL. Rules for converting EER to ODL.
Theory	Object is a triple (i,c,v) where i is unique identifier, c is type constructor and v is object state. A type constructor may be an atom, tuple, set, list, bag and array. There are 2 types of objects-transient and persistent objects. Persistent objects are stored in database and persist after program termination. The typical mechanism for making an object persistent are naming and reachability. The naming mechanism involves giving an object a unique persistent name through which it can be retrieved in this and future programs.
Sample Output	A ODL schema is written with reference to the case study.
Post Lab Assignment	Explain object identity and structure? explain naming mechanism?

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Lab Assignment 3

Lab Assignment 3	
Title	Implement Abstract data type.
Objective	To get familiar User defined data types.
References	<ol style="list-style-type: none">1. Fundamentals of Database Systems Elmasri and Navathe Addison-Wesley Fourth edition2. Database management Systems Raghu Ramakrishnan and Johannes Gehrke TMH3. Database System concepts Korth,Siberchatz,sudarshanMc-Graw-HillFourth edition4. Database Systems,design,Implementation and management Peter Rob and Carlos Coronel Thomson learning. Fifth edition5. Introduction to database systems C.J Date Addison-Wesley Longman
Pre-requisite	Knowledge of →basic data types. →basic DDL commands.
Theory	ADT are the data types created by the programmer having sub-types. An ADT of address may contain may attributes such as street_no, street_name, city, State. Syntax for creating ADT is create or replace type as object (attr1 datatype, attr2 datatype ...) Steps to implement ADT <ol style="list-style-type: none">1. Create ADT for address.2. Create table employee using ADT.3. Describe structure of table and ADT.4. Display all employees staying in mumbai.5. Display address of all employees whose age is greater than 40.6. Display cities of all employees.
Sample Output	create or replace type addr as object(street_no number(3), street_name varchar2(20),city varchar2(10), state varchar2(10)); Type created
Post Lab Assignment	1. write short note on ADT?

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Lab Assignment 4

Lab Assignment 4	
Title	Implement Varrays.
Objective	To study varying array.
References	<ol style="list-style-type: none">1. Fundamentals of Database Systems Elmasri and Navathe Addison-Wesley Fourth edition2. Database management Systems Raghu Ramakrishnan and Johannes Gehrke TMH3. Database System concepts Korth,Silberchatz,sudarshanMc-Graw-HillFourth edition4. Database Systems,design,Implementation and management Peter Rob and Carlos Coronel Thomson learning. Fifth edition5. Introduction to database systems C.J Date Addison-Wesley Longman
Pre-requisite	Knowledge of →ADT.
Theory	<p>V arrays help in storing repeating attributes of a record in a single row. V arrays have a defined lower value 0 and maximum could be any valid number. collector such as varying array allows repetition of only those column values that change potentially storage values. collectors are used to accurately represent relationship between data type in the database object.</p> <p>syntax is create type type_name as varray(5) of datatype</p> <p>Steps to implement Varray</p> <ol style="list-style-type: none">1. Create varray for the item_code.2. Create a table with attributes order_no and item.3. Describe varray structure.4. Display item code for order_no=1 .
Sample Output	create type item_code as varray(5) of varchar2(20); Type created. create table order1(order_no number(5), item item_code); table created.
Post Lab Assignment	<ol style="list-style-type: none">1. Short note on collector andvarray.2. Short note on in-built interfaces.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Lab Assignment 5

Lab Assignment 5	
Title	Implement object Table.
Objective	To study object creation and object structure.
References	<ol style="list-style-type: none">1. Fundamentals of Database Systems Elmari and Navathe Addison-Wesley Fouth edition2. Database management Systems Raghu Ramakrishnan and johannes Gehrke TMH3. Database System concepts Korth,Siberchatz,sudarshanMc-Graw-HillFouth edition4. Database Systems,design,Implementation and management Peter Rob and Carlos Coronel Thomson learning. Fifth edition5. Introduction to database systems C.J Date Addison-Wesley Longman
Pre-requisite	Knowledge of →object identity →object structure
Theory	Each row within the object table has OID-an object identifier value. This is system generated identifier which is being assigned at the time of a new row insertion. create type type_name as object(list of attributes) create table table_name of type_name To view OID of each row select ref(a) from table_name a
Sample Output	select ref(a) from emp a; ref(a) 00002802093B090D8F236BB39 CE 04000
Post Lab Assignment	<ol style="list-style-type: none">1. Explain object structure and object identity?2. Different object structures?

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Lab Assignment 7

Lab Assignment 7	
Title	Implement Member procedure.
Objective	To understand use of objects in member procedure.
References	<ol style="list-style-type: none">1. Fundamentals of Database Systems Elmasri and Navathe Addison-Wesley Fourth edition2. Database management Systems Raghu Ramakrishnan and Johannes Gehrke TMH3. Database System concepts Korth,Siberchatz,sudarshanMc-Graw-HillFourth edition4. Database Systems,design,Implementation and management Peter Rob and Carlos Coronel Thomson learning. Fifth edition5. Introduction to database systems C.J Date Addison-Wesley Longman
Pre-requisite	Knowledge of →PL/SQL
Theory	Creating a member procedure create or replace typename as object (attribute datatype member procedure procedure_name (attribute in datatype) ...); Defining a member procedure create or replace type body typename as member procedure procedure_name(attribute in datatype) is begin Definition partitioning end; end;
Sample Output	step 1: create or replace type T1 as object(ssn number(5),name varchar2(20), member procedure change_name(name1 in varchar2)); Type created step 2: Create table tab of T1; Type created step 3: create or replace type body T1 as member procedure change_name(name1 in varchar2) is begin name:=name1; end; end;
Post Lab Assignment	1. Write short on member procedure.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Lab Assignment 8

Lab Assignment 8	
Title	Implement Member Function.
Objective	To study horizontal Partitioning
References	<ol style="list-style-type: none">1. Fundamentals of Database Systems Elmasri and Navathe Addison-Wesley Fourth edition2. Database management Systems Raghu Ramakrishnan and Johannes Gehrke TMH3. Database System concepts Korth,Siberchatz,sudarshanMc-Graw-HillFourth edition4. Database Systems,design,Implementation and management Peter Rob and Carlos Coronel Thomson learning. Fifth edition5. Introduction to database systems C.J Date Addison-Wesley Longman
Pre-requisite	Knowledge of →features of functions
Theory	Syntax for creating function create or replace objectname as object (attribute1,attribute 2... member function functionname (parameter list) return datatype); CREATE OR REPLACE FUNCTION[SCHEMA.]FUNCTIONNAME (ARGUMENT IN DATATYPE,...)RETURN DATATYPE {IS,AS} VARIABLE DECLARATIONS; CONSTANT DECLARATIONS; BEGIN PL/SQL SUBPROGRAM BODY; EXCEPTION EXCEPTION PL/SQL BLOCK; END;
Sample Output	Create or replace type animal_by as object (breed varchar2(25),name varchar2(25),birthdate date member function AGE(bithdate in date) return number); type created. Create or replace AGE(birthdate date) return number is begin return round(sgs_date_bithdate); end; end; function created
Post Lab Assignment	1 .short note on functions

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Lab Assignment 9	
Title	Implement Partitioning on the tables.
Objective	To study horizontal Partitioning .
References	<ol style="list-style-type: none">1. Fundamentals of Database Systems Elmari and Navathe Addison-Wesley Fouth edition2. Database management Systems Raghu Ramakrishnan and johannes Gehrke TMH3. Database System concepts Korth,Siberchatz,sudarshanMc-Graw-HillFouth edition4. Database Systems,design,Implementation and management Peter Rob and Carlos Coronel Thomson learning. Fifth edition5. Introduction to database systems C.J Date Addison-Wesley Longman
Pre-requisite	Knowledge of →distributed databases →horizontal and vertical partitioning
Theory	Partition improves overall performance but increases overhead on database to search in each partition. Tables are easier to manage. Backup and recovery operations may perform better. Create table table_name(attribute1,attribute2 ...) partition by range(attribute) (partition partitionname values less than(maxvalue),partition partition values less than(maxvalue));
Sample Output	Create table emp(emp_id varchar2(10),name varchar2(20)) partition by range(emp_id) (partition part_id1 values less than(5),partition part_id2 values less than(10)); table created
Post Lab Assignment	1 .write short note on distributed databases.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Lab Assignment 10

Title	Implement XML command.
Objective	To study semi-structured data.
References	<ol style="list-style-type: none">1. Fundamentals of Database Systems Elmari and Navathe Addison-Wesley Fouth edition2. Database management Systems Raghu Ramakrishnan and johannes Gehrke TMH3. Database System concepts Korth,Siberchatz,sudarshanMc-Graw-HillFouth edition4. Database Systems,design,Implementation and management Peter Rob and Carlos Coronel Thomson learning. Fifth edition5. Introduction to database systems C.J Date Addison-Wesley Longman
Pre-requisite	Knowledge of →database →applets →html
Theory	<p>DTD(document type definition) Any valid document conforming to DTD should follow the specified structure:</p> <ol style="list-style-type: none">1. name is given to the root tag and then to the elements and their nested structure are specified.2. a * means element can be repeated 0 or more times3.A + element can be repeated one or more times4.Any element appearing without any of the preceding symbols must appear exactly once in the document. <p>eg</p> <pre><! DOCTYPE projects [<!ELEMENT projects(project+)> <ELEMENT project(pname,number,dept_no,workers)> <ELEMENT pname(#PCDATA)> <ELEMENT number(#PCDATA)> <ELEMENT dept_no(#PCDATA)> <ELEMENT workers(worker *)> <ELEMENT eno(#PCDATA)> <ELEMENT ename(#PCDATA)> ></pre> <p>XSD(XML schema definition) W3 specifies how to formally describe the element in XML document. It helps to verify that each item of content in a document adheres to the description of the element in which the content is to be placed.</p> <pre><xsd:schemaxmnl:sxsd="http://www.w3c.org/2001/XMLSchema"> <xsd:element name="country" type="country" > <xsd:complex type name="country" > <xsd:sequence> <xsd:element name="name" type="xsd:string" > </xsd:sequence> </xsd:complex type > <xsd:schema></pre>
Sample Output	DTD and XSD studied

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Course Description

Title of Course: Seminar

Course Code: MCSE281

L-T-P scheme: 0-2-0

Course Credit: 1

The overall aim of the seminar series is to help develop an emerging field at the intersection of multi-disciplinary understandings of culture and education. It will build on the existing body of work on education and culture, but its aim is explore and develop new perspectives in this area.

The objectives of the six exploratory seminars are:

- **to explore new research from a range of academic disciplines which sheds light on the questions outlined above**
- **to showcase cutting edge research on education and culture from outstanding academic researchers from the UK and internationally**
- **to bring together seminar participants from different disciplines such as Sociology, Philosophy, Psychology, Human Geography, Media Studies as well as Education and Cultural Studies**
- **to encourage and financially support the participation of PhD students**
- **to actively involve practitioners and users from each venue**
- **to engage a core group of policy makers**
- **to use the seminars to develop links between academics and stakeholders in the arts, library, media, community and educational sectors**