Introductory Computing CO103

2-1-0: 3 Credits: 3 Hours Prerequisites: None

Course Outcomes:

1. Understand computing environment, how computers work and the strengths and limitations of computers.
2. Identify and understand the representation of numbers, alphabets and other characters in computer system.
3. Understand, analyse, and implement software development tools like algorithm, pseudo codes and programming structure.
4. Write small programs related to simple/moderate mathematical and logical problems in 'C'.

Course Contents:

Computer Fundamentals: - History, Generations, Classification of Computers; - Organization of a Computer; - Concept of Programming and Programming Languages. Introduction to Programming: - Concept of Algorithm, Flow Chart, Pseudocode, Illustrative Problem Solving Examples. - Features of a Programming Language:
Character Set, Identifiers, Keywords, Data Types, Variables, Declarations, Operators &
Expressions; Statements: Assignment, Input/Output; Flow Control- Conditionals and
Branching; Iteration; Functions, Function Types, Scope Rule; Recursion; Arrays, Pointers, Structures. (A programming language like C/C++ shall be used as a basis language. The same language is to be used for the laboratory).

Text Books:

1. Programming in C, Balaguruswamy.
2. Let us C, Kanetkar Y.
3. Programming in C, Gotfreid, McGrawHill
4. Fundamentals of Computers, Rajaram, V.

Reference Books:

5. The Elements of Programming Style, Kerningham, B. W.
6. Techniques of Program Structures and Design, Yourdon, E.
7. Theory and Problems of Computers and Programming, Schied, F. S.
8. The C Programming Language, Kerningham & Ritchie.

Computing Laboratory	CO104
0-0-2: 2 Credits: 4 Hours	Prerequisites: None

Course Outcomes:

Towards the end of the course the student would:

1. develop knowledge on computations such as algorithm, flowcharts etc.
2. become aware of the existence of various other primitive programming languages that are used for computation.
3. do beginners and average level of programming problems such as factorial, recursion, problems involving use of structures etc.
4. implement some basic data structures using the C language. Some data structures that they can implement using C are Array, Stack, Queue, Linked List etc.

Course Contents:

Laboratory exercises shall involve the following:

1. Familiarization of a computer and the environment and execution of sample programs
2. Expression evaluation
3. Conditionals and branching
4. Iteration
5. Functions
6. Recursion 7. Arrays
8. Structures
9. Linked lists
10. Data structures

It is suggested that some problems related to continuous domain problems in engineering and their numerical solutions are given as laboratory assignments. It may be noted that some of basic numerical methods are taught in the Mathematics course.

Text Books:

1. The Elements of Programming Style, Kerningham, B. W.
2. The C Programming Language, Kerningham & Ritchie.

Reference Books:

3. Programming in C, Balaguruswamy.
4. Let us C, Kanetkar Y.
5. Programming in C, Gotfreid, McGrawHill

Digital Logic Design	CO202
3 - 0 - 1: 4 Credits: 5 Hours	Prerequsites: None

Abstract:

This course is designed to provide knowledge and understanding to sophomores in electrical and computer engineering of Boolean algebra and digital concepts, with concentration on the analysis and design of combinational and sequential logic networks. Furthermore, it provides a foundation for subsequent study in computer organization, architecture, and VLSI design.

Course outcome: Towards the end of the course the student would

1. Have a thorough understanding of the fundamental concepts and techniques used in digital electronics
2. Be able to design, analyze and synthesize logic circuits (combinational & sequential)
3. Be able to design complex digital systems
4. Be able to identify and prevent various hazards and timing problems in a digital system

Course Contents:

History & overview : Reasons for studying digital logic, people who influenced/contributed to the area of digital logic, applications of Digital Logic and introduction to a digital system.

Switching theory: Number systems and codes, Binary arithmetic, Complements, Boolean and switching algebra, Representation and manipulation of switching functions, Minimization of switching functions using algebraic method, K-map(2-,3-,4-,5-variable), Quine McCluskey method.

Combinational logic circuits: Basic logic gates (AND,OR,NOT,NAND,NOR,XOR),
Realization of switching functions with networks of logic gates, 2-level networks: ANDOR,OR-AND,NAND-NAND, NOR-NOR, Multi-level networks, Physical properties of logic gates (technology, fan-in, fan-out, propagation delay), Elimination of timing hazards/glitches.

Modular design of combinational circuits: Design of medium scale combinational logic modules - Multiplexers, demultiplexers, decoders, encoders, comparators, Arithmetic functions (adders, subtracter, carry look ahead), Multipliers, dividers, Arithmetic and logic units (ALUs), Hierarchical design of combinational circuits using logic modules.

Memory elements: Unclocked and clocked memory devices (latches, flip flops), Level vs. edge-sensitive, and master-slave devices, Basic flip flops (SR, D, JK, T), Asynchronous flip flop inputs (preset, clear), Timing constraints (setup time, hold time) and propagation delays, Data registers (selection, clocking, timing), Random-access memory (RAM).

Sequential logic circuits : Finite state machines (FSMs), clocked and unclocked, Mealy vs. Moore models of FSMs, Modelling FSM behaviour: State diagrams and state tables, timing diagrams, algorithmic state machine charts, Analysis of synchronous and asynchronous circuits, Design of synchronous sequential circuits: State minimization, state assignment, next state and output equation realization, Sequential functional units: Data registers, shift registers, counters, sequence detectors, synchronizers, debouncers, controllers.
Fault detection and Location: Fault models for combinational and sequential circuits, Fault detection in combinational circuits; Homing experiments, Distinguishing experiments, machine identification and fault detection experiments in sequential circuits.

Laboratory component:

Study of TTL gate characteristics, Open collector and Tri-state gates, Clock generator and timer circuit.
Synthesis of combinational circuits using NAND, NOR and Multiplexers, Decoder and driver circuits for 7- segment LED displays, D/A converter and 4-bit ALU realization. Synthesis of sequential circuits – study of various types of flip-flops, realization of counters, shift registers and sequence generators.
ASM chart based synthesis such as, Traffic light controller, Blackjack dealer and dice game ASM synthesis, etc.

Text Books:

1. M.M.Mano : Digital Logic and Computer Design, PHI
(EEE) 2. Floyd and Jain : Digital Fundamentals, Pearson Education
3. Switching and Finite Automata Theory, Z. Kohavi, TMH.

Reference Books:

1. R.P.Jain : Modern Digital Electronics, Tata McGraw-Hill Education
2. J.F.Wakerly : Digital Design – Principles and Practices, Pearson Education.
3. Digital Circuits and Logic Design, S. Lee, PHI

Computing Workshop	CO209
0-0-2: 2 Credits: 4 Hours	Prerequisites: None

Abstract:

This course aims to familiarize students with the basic concepts of two programming environments, MATLAB and Python, and their design tradeoffs. The course covers basic programming aspects, and discusses the implementation of MATLAB and Python for different numerical methods. It also summarizes how both these environments provide a wide platform for problem solving and encourages students to explore this course for their upcoming project assignments. A group project has to be taken up by the students during this course.

Course Outcome:

Towards the end of the course the student would:

1. be familiar with basic programming concepts of MATLAB and Python.
2. have a clear understanding of the syntax which will allow him/her to correctly model a problem.
3. be able to design and simulate any real time environment using MATLAB or Python.

Course Contents:

Introduction to MATLAB: MATLAB interface; variables; keywords; commands; operators: arithmetic, relational, logical, bitwise.

Vectors and Matrices in MATLAB: Vectors and matrices: creation, deletion, access, manipulation; Special matrices; complex matrix; matrix commands; matrix operations: determinant, minor, inverse, rank, eigen value and vectors.

MATLAB Scripts: M-files; Function files: primary function, sub-function; ways of creating script files; input/output functions.

Conditional statements in MATLAB: Statements: IF, IF-ELSE, nested IF-ELSE, SWITCH case; IS-function.

Iteration and Loops: Loops: FOR loop, WHILE loop, Nested loops; control statements: break, continue; Vectorizing.

Cell arrays: Creation, deletion, access, manipulation and operations in cell arrays.

Numerical methods using MATLAB: Set operations; Solving of linear equations; Nonlinear equations; differentiation and integration.

Plotting in MATLAB: Visualizing results using plot; subplot; histogram; bar graph; pie chart etc.

Introduction to Python: Python overview; Interactive mode and Script mode; variables; keywords; datatypes: numeric, dictionary, Boolean, set, list, tuple, string; creation, deletion, access in different datatypes; operators: arithmetic, relational, assignment, logical, bitwise, membership, identity; input/output functions.

Conditional statements in Python: Statements: IF, IF-ELSE, nested IF-ELSE.

Iteration and Loops: Loops: FOR loop, WHILE loop, Nested loops; control statements: break, continue, pass.

Functions in Python: arguments: required, keyword, default, variable-length; creating function; return statements.

Matrix in Python: Numpy module for matrix in Python; creation, deletion, access, manipulation; types of matrices; matrix operations: determinant, minor, inverse, rank, eigen value and vectors; solving linear equations.

Plotting in Python: Matplotlib module for plotting in Python: plot, bar graph, pie chart, histogram, scatter plot, contour plot etc.

Textbooks:

1. Steven I. Gordon and Brian Guilfoos, Introduction to Modeling and Simulation with MATLAB and Python, Chapman & Hall, CRC Press, Computational Science Series, 2017.
2. Stormy Attaway, MATLAB: A Practical Introduction to Programming and Problem Solving, College of Engineering, Bosten University, Elsevier, 2009.

References:

1. Mathworks, MATLAB: The Language of Technical Programming.
2. M. C. Brown, Python: The Complete Reference, Mc Graw Hills, 4th Edition, 2018.
3. A. Gilat, MATLAB: An Introduction with Applications, Wiley, 4th Edition, 2012

Data Structures	CO210
3-1-0: 4 Credits: 4 Hours	Prerequisite: CO103, CO104

Course Outcomes:

Towards the end of the course the student would be able to:

1. use and manipulate several core data structures including: Arrays, Linked Lists, Trees, Stacks, Queues for varieties of problems.
2. use critical thinking and problem solving skills in analyzing computational problems from a variety of sources.
3. identify a problem and analyze it in terms of its characteristics and the information needed to solve it.
4. look at different perspectives of the problem, e.g, storage, time complexity

Course Contents:

Time and Space analysis of Algorithms – Order Notations.

Linear Data Structures : Sequential representations – Arrays and Lists, Stacks, Queues, Strings; Link Representations – Linear linked lists, Circular linked lists, Doubly linked lists; Applications.

Recursion – Design of Recursive Algorithms, Tail Recursion.
Nonlinear Data Structures : Trees – Binary Trees, Traversals and Threads, Binary Search
Trees, Insertion and Deletion algorithms, Height Balanced Trees and Weight Balanced Trees, B-trees, B+ trees, Application of trees; Graphs – Representations, Breadth-first and Depth-first Search.

Hashing – Hashing Functions, Collision Resolution Techniques.

Sorting and Searching Algorithms : Bubble sort, Selection sort, Insertion sort, Quick sort, Merge sort, Heap sort, Radix sort.

File Structures: Sequential and Direct Access, Relative files, Indexed files, B+ tree aas index, Multi-index files, Hashed files.

Books:

1. Data Structures and Algorithms, A. V. Aho, J. E. Hoppcroft, J. E. Ullman, Addision Wesley.
2. Fundamentals of Data Structures, E. Horowitzz, S. Sahni, Galgotia Publ.
3. Data Structures using C, A. S. Tanenbaum
4. Algorithms, Data Structures, and Problem Solving, Addision Wesley.
5. Data Management and File Structures, Loomis, Marry, PHI

Data Structures using Object Oriented Programming Lab

0 - 1 - 2 : 3 Credits : 5 Hours

CO211

Prerequsites: CO101, CO102

Course Content:

Review of elementary programming

Recursion: The concept of recursion; recursive specification of mathematical functions (such as factorial and Fibonacci); simple recursive procedures (Towers of Hanoi, permutations, fractal patterns); divide- and-conquer strategies; recursive backtracking; implementation of recursion

Introduction to computational complexity: Asymptotic analysis of upper and average complexity bounds; big-O notation; standard complexity classes; empirical measurements of performance

Fundamental computing algorithms: O(N log N) sorting algorithms (Quicksort, heapsort, mergesort); hashing, including collision-avoidance strategies; binary search trees

Fundamental data structures: Linked structures; implementation strategies for stacks, queues, hash tables, graphs, and trees; strategies for choosing data structures

Object-oriented programming: Object-oriented design; encapsulation and information-hiding; separation of behaviour and implementation; classes, subclasses, and inheritance; polymorphism; class hierarchies; collection classes and iteration protocols; fundamental design patterns

Books:

1. Data Structures and Algorithms, A. V. Aho, J. E. Hoppcroft, J. E. Ullman, Addision Wesley.
2. Fundamentals of Data Structures, E. Horowitz, S. Sahni, Galgotia Publ.
3. Data Structures using C, A. S. Tanenbaum, PHI
4. Herbert Schild: The Complete Reference to C++, Osborne McGrawHill.Bjarne Stroustrup: The C++ Programming Language, Addison Wesley

Data Communication	CO218
3 - 0 - 0 : 3 Credits : 3 Hours	Prerequsites: CO103

Course Outcomes:

1. gain overall idea on the network architecture and issues associated with the different layers.
2. calculate medium capacity for given bandwidth and signal-noise-ratio.
3. understand different analog/digital signals, conversion of signal from one from to another and different modulation techniques.
4. know about different medium access control mechanisms used in telephone/computer network along with relative advantages and disadvantages of one technique over the other.
5. apply error detection/correction techniquesused in data communications.
6. know about different devices associated with datacommunications such as Repeaters, Switches, Routers etc. and their working principles.

Course Contents:

Overview: Objectives and Applications of Computer Communication.

Computer Communication and Network Architecture: ISO-OSI reference model, design philosophy, layer, protocol, interface, and service concepts. Layer wise functionality.

Physical Layer: Concepts of data transmission: signal, communication channel, channel capacity, distortion & noise, line coding; modulation(analog and digital), modem; multiplexing- FDM, TDM, WDM, CDM etc; OFDM & spread spectrum techniques; switching, communication media—guides & unguided; standard protocols, RS- 232C, RS449, X.21, xDSL, SONET, Frame relay, ATM etc.

Medium Assess Control in broadcast networks: ALOHA, CSMA, CSMA/CD, CSMA/CA, token ring, token bus etc.

Standard LAN Protocols: (IEEE 802.X), FDDI, satellite networks, LAN switching, VLAN, WLAN, PAN and WiMax.

Data link layer: Framing, Error control techniques, Data link protocols and their performance, HDLC and PPP protocol.

Network layer: Introductory concepts and issue: Routing, Congestion and deadlock control Algorithms, Internetworking issues and devices, gateways, bridges and routers, IP & X.25 protocols.

Communication Laboratory:

Laboratory: Generation, testing, of AM, FM, and PM, Transmitter and receiver, PCM codec; Flow control, Error Control and MAC protocols on LAN trainer kit.

Books:

1. Stalling, Data and Computer Communication, 8e, PHI (EEE)
2. Tanenbaum A.S., Computer Network, 5e, PHI (EEE)

References:

1. Forouzan B. A, Data Communication and Networking, 5e, Tata McGrawHill
2. Leon-Garcia, Widijaja, I., Communication, 5e, PHI (EEE)
3. B. P. Lathi, "Modern Analog and Digital Communication Systems", 3/e, Oxford University Press, 1998.

Computer Architecture & Organization CO214

3 - 0 - 1 : 4 Credits : 5 Hours Prerequsites: CO 103, CO104

Course Outcomes:

1. be well conversant in the concepts of computer architecture and organization for several engineering applications.
2. be able to analyze and design different memory organizations.
3. be well equipped with the knowledge of Cache Mapping techniques and Virtual memory.
4. be able to demonstrate programming proficiency using the various addressing modes and data transfer instructions of the target computer.

Course Contents:

Basic organization of the computer and block level description of the functional units from program execution point of view; Fetch, decode and execute cycle; Assembly language programming: Instruction set, instruction cycles, registers and storage, addressing modes; discussions about RISC versus CISC architectures;
Inside a CPU: information representation, computer arithmetic and their implementation; control and data path, data path components, design of ALU and data path, control unit design;
Memory and I/O access: Memory maps, Read Write operations, Programmed I/O, Concept of handshaking, Polled and Interrupt driven I/O, DMA data transfer; I/O subsystems: Input-Output devices such as Disk, CD- ROM, Printer etc.;
Interfacing with
IO devices, keyboard and display interfaces;
Inside the Memory: memory organization, static and dynamic memory; Cache memory and Memory Hierarchy – Cache memory access techniques; Virtual memory;
Introduction to Parallel Architectures: Instruction Level Parallel Processors- Pipelined, VLIW, Superscalar; Multiprocessors & Multicomputer Architectures, Vector Processing.

Laboratory experiments:

The assignments should cover the following:

1. Assignments on assembly language programming;
2. Experiments on synthesis / design of simple data paths and control unit;
3. Assignments on interfacing devices and systems like data acquisition systems ; Development kits as well as PCs/Workstations may be used for the laboratory, along with design / simulation tools as and when necessary.

Books:

1. Computer Architecture and Organization, Hayes J. P., McGrawHill 2. Computer Organization, Hamacher, Zaky, Vranesic, McGrawHill
3. Computer System Architecture, Mano M. M.

Computer Organization Lab	CO215
0 - 0 - 1 : 1 Credits : 2 Hours	Prerequsites: CO103

Abstract:
This course is comprised of a laboratory component that is to be covered in parallel to Computer Architecture and Organization (CO212). The course is aimed at providing a practical understanding on building of the functional units and their integration and for appreciation of the issues on programming at the machine level.

Course Outcome:

1. Practical understanding of the architectural aspects of a computer at the machine level
2. Getting equipped with practical aspects of the working of a computer that will be useful for courses such as Operating Systems, Embedded Systems etc.
3. Ability to work on development of digital systems.

Course Content:

Circuit Design and Simulation: Register, Counter, Adder, Multiplier, Data Paths, Control Unit, ALU etc.
Introduction to 8086 family microprocessors, Architecture of 8086 - Register set, Concept of segments, use of the register set, use of stack, Instruction set, PSW Flags.
8086 assembly language programming: Software interrupts, Data Input/Output, Arithmetic Operations, String Handling, Branching, Looping, showing conditional and unconditional branches, and LOOP instruction, Creating Subroutines.

List of Laboratory Assignments:

Set 1: Designing CPU components using a Simulator:

- Register, Counter Adder, Multiplier
- Data Paths, Control Unit
- ALU, Memory unit

Set 2: 8086 Assembly Language Programming:

- Taking keyboard input for characters and numbers, Displaying and working with multi-digit number, Arithmetic operations
- Comparison operators, conditional and unconditional jumps, loops
- Working with array of numbers and strings of characters, finding average/mean of numbers, searching
- Writing Procedures, passing and returning values, use of stack

Text Books:

1. Computer System Architecture, Mano M. M, Pearson.

2. Guide to Assembly Language Programming in Linux, Sivarama Dandamudi, Springer

Reference Books:

1. IBM PC Assembly Language and Programming, Peter Abel, 3e, PHI.
2. Computer Organization and Design: The Hardware/ Software Interface, Patterson and Hennessy, Elsevier.
3. Computer Organization and Architecture: Designing for Performance 9E, William Stallings, Pearson Education.
4. Computer Organization, Hamacher, Zaky, Vranesic, McGrawHill.

Formal Languages & Automata CO216

3 - 0 - 0 : 3 Credits : 3 Hours Prerequsites: CO101

Course Outcomes:

1. understand the notion of effective computability.
2. emphasize the engineering applications of the theory developed.
3. appreciate the central issues by semi-formal intuitive reasoning.
4. develop the ability to apply the ideas and proof techniques in varied environments

Course Contents:

Alphabet, languages and grammars.

Production rules and derivation of languages.

Chomsky hierarchy of languages.

Regular grammars, regular expressions and finite automata (deterministic and nondeterministic). Closure and decision properties of regular sets. Pumping lemma of regular sets. Minimization of finite automata.

Left and right linear grammars. Context free grammars and pushdown automata.

Chomsky and Griebach normal forms. Parse trees, Cook, Younger, Kasami, and Early's parsing algorithms.

Ambiguity and properties of context free languages. Pumping lemma, Ogden's lemma, Parikh's theorem.
Deterministic pushdown automata, closure properties of deterministic context free languages.

Turing machines and variation of Turing machine model, Turing computability

, Type 0 languages. Linear bounded automata and context sensitive languages.

Primitive recursive functions. Cantor and Godel numbering. Ackermann's function, murecursive functions, recursiveness of Ackermann and Turing computable functions.

Church Turing hypothesis. Recursive and recursively enumerable sets. Universal Turing machine and undecidable problems. Undecidability of post correspondence problem. Valid and invalid computations of Turing machines and some undecidable properties of context free language problems.

Books:

1. J. E. Hopcroft and J. D Ullman: Introduction to Automata Theory, Languages and Computation, Addison Wesley Publ., New York.
2. McNaughton R, Elementary Computability, Formal Languages and Automata, PrenticeHall.
3. Martin J C, Introduction to Languages and the Theory of Computation, McGraw-Hill International Edition.

References:

1. Buchi A, Finite Automata, Their Algebras and Grammars: Towards a Theory of Formal Expressions, Springer-Verlag.
2. H. R. Lewis and C. H. Papadimitriou: Elements of the Theory of Computation, Prentice Hall, Englewood Cliffs.
3. F. Hennie: Introduction to Computability, Addison Wesley Publ., New York.

Design and Analysis of Algorithms CO206

3 - 0 - 1 : 4 Credits : 5 Hours Prerequsites: CO203

Course Outcomes:

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

1. compare, contrast, and apply the key algorithmic design paradigms: brute force, divide and conquer, decrease and conquer, transform and conquer, greedy, dynamic.
2. compare, contrast, and apply key data structures: trees, lists, stacks, queues, hash tables, and graph representations.
3. define, compare, analyse, and solve general algorithmic problem types: sorting, searching, string processing, graphs, and geometric algorithms.
4. implement, empirically compare, and apply fundamental algorithms and data structures to real-world problems.
5. compare, contrast, and apply algorithmic trade offs, get an introduction to the world of complexity theory.

Course Contents:

Algorithms and Complexity – asymptotic notations, orders, worst-case and average-case, amortized complexity.

Basic Techniques – divide & conquer, dynamic programming, greedy method, backtracking, branch and bound, randomization.

Data Structures – heaps, search trees, union-find problems.

Applications – sorting & searching, combinatorial problems, optimization problems, computational geometric problems, string matching.

Graph Algorithms – BFS and DFS, connected components, spanning trees, shortest paths, max-flow. NP-completeness.

Approximation algorithms.

Laboratory: The laboratory component will emphasize two areas:

Implementation of algorithms covered in class: This will involve running the algorithms under varying input sets and measuring running times, use of different data structures for the same algorithm (wherever applicable) to see its effect on time and space, comparison of different algorithms for the same problem etc.

Books:

1. Introduction to Algorithms, Cormen et al., McGrawHill
2. Aho A, Hopcroft J., Ullman J., The Design and Analysis of Algorithms, Addison- Wesley.

Graph Theory	CO217
3-0-0: 3 Credits: 3 Hours	Prerequisites: CO210

Course Outcomes:

1. understand clearly the basic theories of Graph Theory.
2. map a given real life problem to a graph theory problem.
3. apply graph theory concepts in solving real life problems.
4. design and/or implement algorithms for solving some common graph theory problem.

Course Contents:

Graph : Incidence and degree; Handshaking Lemma; Isomorphism; Subgraphs and Union of graphs; Connectedness; Walks, Paths and Circuits; Components and Connectedness; Walks, Walks, Paths and Circuits; Components and Connectedness algorithms; Shortest Path Algorithms, Eulerian graph, Fleury's algorithm and Chinese postman problem; Hamiltonian graph - necessary and sufficient conditions; Traveling salesman; Bipartite graph.

Tree: Properties of trees; Pedant vertices in a tree; Center of a tree; Rooted binary trees; Spanning trees, Spanning tree algorithms; Fundamental circuits; Spanning trees of a weighted graph; cut-sets and cut- vertices; Fundamental cut-sets; Connectivity and separativity; network flow; max-flow min-cut theorem.

Planner graph: Combinatorial and geometric dual; Kuratowski's graph; detection of planarity; Thickness and crossings.

Matrix representations of graph: Incidence; Adjacency; matrices and their properties.

Colourings: Chromatic number : Chromatic polynomial; The six and five colour theorems; The four colour problem.
Directed graphs: Binary relations; Directed graphs and connectedness; directed trees; Aborecence; Polish method; Tournaments.

Counting of labeled trees: Cayley's theorem; Counting methods; Polya theory.

Books:

1. Deo, N.: Graph Theory with Applications to Engineering and Computer Science.
2. Harary : Graph Theory, PHI (EEE)

Operating Systems CO309

3-0-0: 3 Credits: 3 Hours Prerequisites: CO210 (Data Structure), CO214(CAO)

Abstract: The course CO309 Operating Systems intends to build the basic concepts of the operating system software and its implementation details for computer system. It also covers topics with case studies giving the students opportunity to explore practical operating systems like - MS Windows, Unix, Linux. The course will be instrumental to build the confidence in the students to develop the introductory knowledge of design and develop an operating system.

Course Outcomes:

Towards the end of the course the student would be conversant with

1. Operating systems and OS design issues
2. OS based programming fundamentals
3. be familiar with OS system software design concepts
4. be able to design and develop OS features
5. OS based application development

Course Contents:

Overview: Evolution of Operating Systems, current status and future trends. Structural overview, system calls, functions of OS, Hardware requirements: protection, context switching, privileged mode

Concept of a process: states, operations with examples from UNIX/Linux (fork, exec) and/or Windows. Process scheduling, interprocess communication (shared memory and message passing), UNIX/Linux signals, cooperating and concurrent processes, tools, and constructs for concurrency,

Threads: thread management, multithreaded model, scheduler activations, examples of threaded programs and applications.

Scheduling: multi-programming and time sharing, scheduling algorithms, multiprocessor scheduling, thread scheduling (examples using POSIX threads).

Process synchronization: mutual exclusion, shared data, critical sections, classical two process and n-process solutions, hardware primitives for synchronization, lock, semaphores, monitors, block and wakeup, classical problems in synchronization (producer-consumer, readers-writer, dining philosophers, etc.).

Deadlocks: modeling, characterization, prevention and avoidance, detection, and recovery.

Memory management: with and without swapping, MMU, Contiguous and noncontiguous allocation, paging and segmentation, demand paging, virtual memory, page replacement algorithms, working set model, thrashing, and implementations from operating systems such as UNIX, Windows. Current Hardware support for paging: e.g., Pentium/ MIPS processor etc.

Secondary storage and Input/Output: device controllers and device drivers, disks, scheduling algorithms, file systems, directory structure, device controllers and device drivers, disks, disk space management, disk scheduling, NFS, RAID, other devices and operations on them, UNIX FS, UFS protection and security.

Virtualization: Virtual Machine (VM), concept of hypervisor and virtual machine manager (VMM), types of hypervisor: kernel-based and hosted hypervisor, open source virtual machine design in Linux: Kernel-based Virtual Machine (KVM).

Protection and security: Illustrations of security model of UNIX and other OSs. Examples of attacks. Pointers to advanced topics (distributed OS, multimedia OS, embedded OS, real-time OS, OS for multiprocessor machines, mobile OS, cluster OS).

Case study: Design of UNIX, Linux, Windows, Android

Textbooks:

1. Operating System Concepts, Abraham Silberschatz, Peter B. Galvin, Greg Gagne, 9th Ed., John Wiley, 2018.
2. Operating Systems: Internals and Design Principles, William Stallings, Pearson, 9th Ed., 2019.

Reference Books:

1. Operating systems: Concepts and Design, M. Milenkovic (McGraw Hill, 2001)
2. Modern Operating Systems, A. S. Tanenbaum (Pearson, 2009)
3. Design of the Unix Operating System, M. J. Bach (Prentice Hall of India, 1986)
4. Operating Systems: Three Easy Pieces, Remzi and Andrea Arpaci-Dusseau
5. http://pages.cs.wisc.edu/~remzi/OSTEP/
6. Understanding Linux Kernel 2.6, Bovet and Chesti (Orelly), 2005)
7. Professional Linux Kernel Architecture, Wolfgang Mauerer, (Wiley)
8. Linux Kernel Development, R. Love (Addison Wesely, 2010)
9. Professional Android Application Development, R. Meier (John Wiley & Sons)

Software Engineering	CO311
3-0 -0: 3 Credits: 3 Hours	Prerequisites: CO209(CW)

Abstract: This course is to make students familiar with overall concepts involved in the software development process including the current scenario prevailing in the software industry. The course covers topics on software development process such as feasibility study, requirement specification, different facets of software design, coding, testing etc. The course also covers project planning and management.

Course Outcomes:

After the course, a student will be:

1. Familiar with overall concepts involved in the software development project.
2. Able to understand fundamental concepts of requirements engineering and Analysis Modelling.
3. Able understand the various software design methodologies
4. Able to understand various testing and maintenance measures.
5. Able to understand the process of project planning and management techniques.

Course Contents:

Module I: Introduction to software engineering

Evolution and Impacts of Software Engineering, Software life cycle models and their comparative study

Module II: Requirements analysis and specification

Software requirements, Software requirements engineering, Requirements specifications techniques. Formal requirements specification and verification - axiomatic and algebraic specifications

Module III: Software design

Basic Issues in software design- modularity, cohesion, coupling and layering. Design approaches- top-down & bottom-up, Function-oriented software design, data flow diagram and structured charts, Object-oriented design, Object modelling using UML, Use case model development, Design specification and notations

Module IV: Software implementation

Structured coding techniques, coding styles, and standards; Guidelines for coding and documentation, automatic code generation.

Module V: Software verification, validation, and maintenance

Theoretical foundation; black box and white box approaches; Integration and system testing, Static and Dynamic Analysis tools, Software Maintenance – Types, maintenance models, reverse and forward Engineering, Maintenance Cost models, Computer Aided Software Engineering (CASE)

Module VI: Software reliability

Definition and concept of reliability; software faults, errors, repair, and availability; reliability and availability models.

Module VII: Software Project Management

Project Management, Project planning and control, Cost estimation and evaluation techniques, cost estimation based on COCOMO model and Raleigh model; Project Scheduling using PERT and GANTT charts, Organizational structure planning, project formats and team structures; Risk analysis and planning, Software configuration management

Text Books:

1. Rajib Mall, Fundamentals of Software Engineering, Prentice Hall India Learning Private Limited (Fifth Edition, 2018)
2. Ian Sommerville, Software Engineering, Pearson Education, 2017

References:

1. R.S. Pressman, Software Engineering: A Practitioner's Approach, McGraw Hill, 2014
2. Robert C Martin, Clean Code, Pearson Education, 2012
3. Steve McConnell, Code Complete, Dreamtech Press, 2011
4. Gregor Hohpe, Enterprise Integration Patterns, Pearson, 2011
5. Martin Fowler, Patterns of Enterprise Application Architecture, Pearson, 2012

Operating Systems Lab	CO310
0-0-1: 1 Credit: 2 Hours	Prerequisites: CO309, CO103 (IC), CO104 (CL), CO210 (Data Structure)

Abstract: The course CO310 Operating System Lab intends to build hand on understanding about some of the important topics covered in CO309 course. It will familiarize with UNIX system calls for process management and inter-process communication, process synchronization, memory management and file system management; experiments on process scheduling and other operating system tasks through programming, simulation / implementation under a simulated environment.

Course Outcomes:

After completion of course, students would be able

1. to grasp knowledge for implementation of operating system software features,
2. to understand the working of an operating system kernel,
3. to master on using system level programming especially the kernel coding using C and java /C++ languages,
4. to master on develop algorithm for operating system design
5. to grasp knowledge for implementing file system concepts
6. to master in using simulation tools, system utilities

Course Contents:

Shell scripting primer using Bourne shell, Bash scripting for beginner, use of awk etc.

Create process (use of fork(), exec() etc. system calls), implement a process ownselves

Use system calls signal(), kill(), creating POSIX threads, using thread library Pthread library using system calls pthread_create() and pthread_exit()

Implementation of file locks using fcntl for basic file access synchronization

Use of basic IPC mechanism with pipe(), mknod(), using message queue, shared memory.

Learn to use synchronization of processes with semaphore and other tools,

Dynamic memory allocation, LKM programming, Device driver for char and block devices

Open source Linux kernel source code browsing and understanding.

Android based application development.

Open source hypervisor development

Reference Books:

1. Unix Network Programming, Stevens, Vol-1, Addison-Wesley, 3rd Ed, 2003 & Vol2, Prentice Hall, 2nd ed, 1998

2. Design of Unix Operating System, M. J. Bach (Prentice Hall of India, 1986)
3. Linux Device Drivers, J. Corbet and A. Rubini, Orelly, 2005
4. Professional Android Application Development, R. Meier (John Wiley & Sons, 2014)
5. Writing a simple Operating System from Scratch, N. Blundell, University of Birmingham, UK, 2010
6. http://www.ee.surrey.ac.uk/Teaching/Unix/
7. http://developer.android.com/guide/index.html
8. Operating Systems: Three Easy Pieces, Remzi and Andrea Arpaci-Dusseau http://pages.cs.wisc.edu/~remzi/OSTEP/
9. Web links and tutorial materials will be provided time to time

Database Systems	CO312
3-0-0: 3 Credits: 3 Hours	Prerequisites: CO210 (DS)

Abstract: The course Database System is an introductory course on database systems. The course covers the basic concepts of database, data models, database architecture, relational database languages, SQL, functional dependency and normalization, database transactions

Course Outcomes:
After completion of this course:

1. The students should clearly understand the concepts of database systems, their advantages, and applications.
2. The students should be able to design a database system for a given database problem.
3. The students should be familiar with the emerging database application areas.

Course Contents:

Introduction & Overview: Concept of database, Characteristics of database, Advantages, data independence, redundancy Control; Database architecture - ANSI model.

Modelling of real-world situation (data models): ER model, EER model

Relational data model: relational model concepts, relational algebra and calculus, SQL, ER/EER to relational model mapping,

Functional dependencies and normalization: functional dependencies, normal forms, decomposition, multi-valued functional dependency, and higher normal forms

Database Indexing and hashing: B-Tree, B+ Tree, static and dynamic hashing

Database Transaction concepts, query evaluation overview, security, and recovery

Distributed Database

Brief introduction to emerging database applications (like Hadoop, NoSQL etc.)

Text Books:

1. Fundamentals of Database Systems, Sixth(2011)/Seventh(2017)Edition, ELMASRI and NAVATHE, Pearson
2. Database Systems Concepts, Sixth (2010)/Seventh(2019) Edition, A. SILBERSCHATZ, H. F. KORTH, S SUDARSHAN, McGraw Hill,

References:

1. Database Management Systems, 3rd Edition, - Raghu Ramakrishnan, Johannes Gehrke, McGraw Hill, 2014
2. An Introduction to Database Systems, 8th Edition, C.J Date, Pearson, 2003
3. Fundamentals of Database Systems, by Leon & Leon, Tata McGraw Hil, 2008
4. SQL & NoSQL Databases, Meier, Andreas and Kaufmann, Michael, eBook, Springer 2019

5. Getting Started with NoSQL, Gaurav Vaish, Packt Publishing, March 2013
6. Hadoop: The Definitive Guide, 4th Edition, Tom White, O'Reilly Media, Inc., 2015

Database Systems Lab	CO313
0 - 1 - 1: 2 Credits: 3 Hours	Prerequisites: CO210

Course Outcomes:

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

1. Create and manage a database using RDBMS like Oracle.
2. Perform queries on the database.
3. Build user-defined functions and procedures using PL/SQL.
4. Develop database applications.

Course Contents:

Introduction: Introduction to a RDBMS like Oracle

SQL: data types, DDL, DML

SQL functions, PL/SQL and user-defined function, triggers
DBA commands

Role-based authorization

Development of database applications using tools/languages like Forms & Reports, PHP, Java, JavaScript etc.

Laboratory works / experiments:

- Basic commands of SQL and SQL Plus
- Creating and modifying database tables, inserting, deleting and updating data in database tables using SQL
- SQL commands for retrieving data from database tables
- Creating and updating database triggers
- Writing and executing SQL scripts
- SQL scripts for building simple reports
- Basics of PL/SQL
- User-defined functions and procedures
- DBA commands and database authorization
- Developing GUIs using Oracle Forms & reports/PHP/Java etc.
- Developing reports using Oracle Forms & reports/PHP/Java etc.

Text Books:

1. SQL The Complete Reference, 3rd Edition, James Groff, Paul Weinberg, McGraw Hill Education, 2017
2. Oracle 12c: The Complete Reference, Kevin Loney, George Koch McGraw Hill/Osborne, 2017
3. Learning PHP, MySQL & JavaScript 5e (Learning PHP, MYSQL, Javascript, CSS & HTML5), Robin Nixon, O'Reilly, 2018

Reference Books:

1. Beginning PHP and Oracle, W. J. Gilmore and B. Bryla, Apress, Berkley, CA, USA, 2007
2. The Underground PHP and Oracle Manual, Release 2 Christopher Jones and Alison Holloway, Oracle, 2012
3. Mastering Oracle SQL, 2nd Edition Sanjay Mishra and Alan Beaulieu, O'Reilly Media, 2004
4. Oracle PL/SQL Programming, 5/6th Edition, By Steven Feuerstein, Bill Pribyl, O'Reilly Media, 2016
5. Oracle Forms developer's handbook, Albert Lulushi, Pearson Education, 2012
6. Oracle 9i development by example, Dan Hotka, Pearson Education, 2001

Computer Graphics	CO303
3-0-1: 4 Credits: 5 Hours	Prerequisites: CO210

Abstract: This course aims to familiarize students with drawing images while learning the underlying algorithms for two and three dimensional graphics and working with animations on the computer. This course is programming intensive starting from basic drawing on the computer to special effects in movies.

Course Outcomes:

This course will give the students hands-on experience at developing interactive, real-time rendering applications using C/OpenGL. At the end of the semester the course outcomes would be:

1. Designing and implementing an application which illustrates the use of various rasterization and transformation techniques.
2. Studying, comparing, and implementing various methods for computer representation of objects.
3. Animating the smooth motions of objects in a scene and predicting collisions between the implemented objects.

Course Contents:

Display Devices: Line and point plotting systems; raster, vector, pixel and plotters, Continual refresh and storage displays, Digital frame buffer, Plasma panel displays, Very high resolution devices, High-speed drawing, Display processors, Character generators, Colordisplay techniques (Shadow-mask and penetration CRT, analog false colors, hard-copy color printers).

Display description: Screen co-ordinates, user co-ordinates; Graphical data structures (compressed incremental list, vector list, use of homogeneous co-ordinates); Display code generation; Graphical functions.

Output Primitives: Line drawing algorithms, Circle and Ellipse generating algorithms, Other curves & Conic sections, Polynomials and spline curves.

Filled area primitives: Scan-line polygon fill algorithm, Inside-outside tests, Boundary fill algorithm, Flood fill algorithm, Character generation.

Attributes of output primitives: Line attributes, Curve attributes, Color and grayscale levels, color tables, Area fill attributes: fill styles, Character attributes, Antialiasing.

2D geometric transformations: Basic transformation: translation, rotation, scaling, Composite transformations, Reflection and shearing, Transformations between coordinate systems, Affine transformations.

2D viewing: Viewing pipeline, window-to-viewport coordinate transformation, Clipping operations, Point clipping, Line clipping algorithms, Polygon clipping algorithms, Curve clipping, Text clipping

Interactive Graphics: Pointing and positioning devices (cursor, light pen, digitizing tablet, the mouse, track balls). Interactive graphical techniques; Positioning, Elastic

Lines, Inking, Zooming, Panning, Clipping, Windowing, Scissoring. Basic positioning methods.

3D Concepts: 3D display methods: Parallel & perspective projection, Depth cueing, Visible line and surface, identification, Exploded and cutaway views, 3D and stereoscopic views, Polygon surfaces, tables, equations, meshes, Curved lines and surfaces. Quadric surfaces, sphere, ellipsoid, torus, superellipse, superellipsoid, Spine representations, Bezier, cubic Bezier curves and surfaces, Sweep representations, Octrees & BSP trees, Fractals.

3D transformations and Viewing: 3D transformations & composite transformations, Viewing pipeline, viewing coordinates, Wire-frame perspective display, Perspective depth, Projective transformations
Visible surface detection methods: Back-face detection, A-buffer method, Scan-line method, Depth-sorting method, BSP-tree method, Octree methods, Ray casting and wireframe methods

Illumination models and surface rendering: Basic illumination models, specular reflection and Phong model, Hidden line and surface elimination, Transparent solids, Shading, halftone patterns and dithering, Ray tracing, Texture mapping

Animation: Animation sequence designing, key framing, morphing, simulated accelerations, motion specifications.

Computer Graphics using OpenGL: An introduction to OpenGL basic graphics primitives, Transformations using OpenGL, Drawing 3D scenes with OpenGL, Introduction to Rendering methods (with various shaders - vertex shader, fragment shader).

Text Books:

1. John F. Hughes, Andries Van Dam, Morgan Mc Guire ,David F. Sklar , James D. Foley, Steven K. Feiner and Kurt Akeley, “Computer Graphics: Principles and Practice”, , 3rd Edition, Addison- Wesley Professional,2013.
2. Edward Angel, Interactive Computer Graphics: A Top-Down Approach with OpenGL, 4th edition, Addison-Wesley, 2005.

Reference Books:

1. Donald Hearn and M. Pauline Baker, Warren Carithers,“Computer Graphics With Open GL”, 4th Edition, Pearson Education, 2010.
2. Peter Shirley, Michael Ashikhmin, Michael Gleicher, Stephen R Marschner, Erik Reinhard, KelvinSung, and AK Peters, Fundamental of Computer Graphics, CRC Press, 2010.

System Software and Compiler Design CO314

3-0-1: 4 Credits: 5 Hours Prerequisites: CO214, CO216

Abstract: The course CO314 System Software and Compiler Design gives an idea of system software in general, and programming tools. It provides an overview of text editors, translators, linkers, loaders, debugger, and some common text processing tools. The course includes design and implementation of assemblers and compilers and macro processors. The task of compilation is an elaborate one with multiple phases and many theoretical aspects.
This course includes suitable modelling and mechanisms for this task and some useful tools. The theoretical concepts of translators are complemented by adequate practical exercises. This course also covers the concept of program linking, relocation and loading, and the functions and features of text editors, debuggers, and some common text processing tools.

Course Outcomes:

Towards the end of the course the student would understand-

1. The task of translation of assembly language and high-level language programs into machine language programs,
2. Description of languages using formal grammars and use of tools for recognition of input strings.
3. Automate the task of parser construction and produce translations of input.
4. Implement symbols tables.
5. The run-time memory and execution environment of programs.
6. The task of code optimization.
7. The scope of a compiler in leveraging the advances of computer architecture.
8. The various types of linking of program modules.
9. The tools available for program debugging, version control, building large executable programs.
10. The common tools for text search and simple editing.

Course Contents:

Overview: Definition and classification of system software, System software for program creation- editors, programming language translators, linkers and loaders, debuggers.

Introduction to machine language, assembly language and high-level language.

Assemblers: Outline of an assembler Lexical analysis: specification of tokens, regular expressions, token recognition Assembler data structures, single pass and multi-pass assemblers, Assembler macros and macro-processors.

Compilers: Characteristics of High Level Languages (HLL), Outline of a compiler; Syntactic specification of HLL using CFG, ambiguity.

Parsing: Parse trees and derivations, top-down parsing, recursive descent parser; bottom-up parsing, Operator precedence parser, LR parsers- SLR, CLR and LALR, handling ambiguity Syntax directed translation, Intermediate Code generation, Semantic analysis, Attribute grammars, Type checking, type conversion and overloading; Symbol tables for compilers.

Runtime environments: activation records, heap management, garbage collection Error detection and recovery.

Code optimization: basic blocks, directed acyclic graph representation, local and global optimization, data flow analysis, register allocation, Loop optimization, Instruction Scheduling and Software Pipelining, Automatic Parallelization.

Text editors: Basic features of a text editors, data structures for text editors.

Linkers and loaders: Basic concepts, Relocation, Static and Dynamic linking, shared libraries, loaders, overlays.

Debugger: features, case study: gdb (laboratory practice)

Unix Utilities: make, RCS, grep, sed (laboratory practice).

Practical Topics:

1. Familiarization with text files and binary file operations.
2. Implementation of lexical analyser. Use of tool flex.
3. Implementation of a two-pass assembler
4. Implementation of a simple desk calculator
5. Implementation of operator precedence parsing
6. Representation of context free grammar in data structure and algorithm for creation of an operator precedence table.
7. Implementation of recursive descent parsing
8. Algorithm for creation of LL(1) parsing table.
9. Use of tool bison for creating an LALR parser.
10. Use bison for creating three address code for simple program segment and a symbol table.
11. Use of gdb debugger.
12. Use of tools make, RCS, grep, sed
13. Familiarization of object code format and linker in Linux.

Text Books:

1. D M Dhandhere. System programming and operating systems, 2e. Tata McGraw Hill, 1999.
2. Alfred Aho, Jeffrey Ullman, Monica S. Lam, and Ravi Sethi. Compilers: Principles, Techniques, and Tools, 2e. Pearson, 2007.

Reference Books:

1. Andrew W Appel. Modern compiler implementation in Java, 2e. Cambridge, 2009.
2. Sumitabha Das, Unix System V.4 Concepts and Applications, TMH.
3. Linux Manuals.
4. Windows Manuals.

Computer Networks	CO315
3-0-0: 3 Credits: 3 Hours	Prerequisites: CO218 (Data Communication)

Abstract: CO315 is an introductory course in the field of computer network at Tezpur University. This course assumes that the students have the prerequisite knowledge of data communications, where students have learned basic networking concepts up to the medium access control sublayer. This course covers the basics of TCP/IP protocol stacks starting from Network Layer and going up to the different applications of computer networks. It is designed to teach the students the very basics of computer networks as an infrastructure and to make them understand the needs of different kinds of applications which are designed to run on this infrastructure.

Course Outcomes:

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

1. Understand the need of a layered Architecture for Computer Network.
2. Understand the key functions performed by different widely known protocols at Network, Transport and Application Layers.
3. Configure a network with Router, ARP, DHCP, DNS, and Gateway etc.
4. Apply mathematical foundations to solve computational problems in computer networking
5. Understand the basic working behavior of TCP and UDP.
6. Know the use of different networking devices.
7. Understand the needs of network security and usages of different security mechanisms.
8. Understand the needs and protocols used in different network applications
9. Understand the basic mechanisms used in data compression

Course Contents:

Review of computer network architecture and the subnet layers.

Network layer: Design issues, Addressing, Routing, internetworking issues, Internet Protocol, IPv4, IPv6, ARP, DHCP, ICMP, Routing algorithms: Distance vector, Link state, Metrics, Inter-domain routing. Subnetting, Classless addressing, CIDR based routing and forwarding, Network Address Translation, Mobile IP, MPLS, VPN.

Introductory queuing theory.

Transport layer: Design issues, UDP, TCP. Connection establishment and termination, sliding window, flow and congestion control, timers, retransmission, TCP extensions (Tahoe, Reno, New-Reno, SRP etc.), RPC, RTTP

End-to-end Data : Presentation formatting issues and methods: XDR , ASN.1, NDR; data compression, lossless compression algorithm, run length encoding, DPCM, Huffman coding, dictionary-based methods: LZ77, LZ78 and their variations, image compression- JPEG, video compression- MPEG, newer compression standards;

Network Security: Concepts of symmetric and asymmetric key cryptography. Sharing of symmetric keys - Diffie Hellman. Public Key Infrastructure. Public Key

Authentication Protocols. Symmetric Key Authentication Protocols. Pretty Good Privacy (PGP), IPSec, Firewalls.

Application: Client-Server and Peer-to-Peer applications, Application layer examples: DNS, SMTP, IMAP, HTTP, P2P systems, BitTorrent, network file system, network management.

New trend in Networking: SDN, IoT, Cloud etc.

Text Books:

1. AS Tanenbaum and DJ Wetherall, Computer Networks, 5th Ed., Pearson, 2016.
2. LL Peterson and BS Davie, Computer Networks: A System Approach, 5th Ed.,Morgan Kaufmann Publishers Inc., 2011

References Books:

1. JF Kurose and KW Ross, Computer Networking: A Top-Down Approach featuring the Internet, 3rd Edition, Pearson Education, 2017.
2. K Sayood, Introduction to data Compression, 5th Ed., Morgan Kaufmann, 2020
3. WR Stevens, UNIX Network Programming, 2nd Ed., 2015, Pearson
4. DE Comer and DL Stevens, TCP/IP Programming Vol. I, II , III, 2nd Ed, 2015, Pearson.

Computer Networks Lab	CO316
0-0-1: 1 Credit: 2 Hours	Prerequisites: CO218 (DC)

Course Outcomes:

At the end of this course the student should be able to:

1. Apply packet sniffing tools to capture packets and analyse
2. Write client-server applications in C/C++/Java
3. Use different OS tools to configure network and network protocols
4. Set up LAN using networking devices
5. Use network simulators to study behaviour of different protocols
6. Analyse performance of various communication protocols using packet sniffers

Course Contents:

Experimental study of application protocols such as HTTP, FTP, SMTP, using network packet sniffers and analyzers such as Ethereal, tcpdump, Wireshark etc.

Client-server based application development using socket programming in C/C++/Java (both TCP and UDP sockets)

Experiments with packet sniffers to study the behaviour of TCP protocol.

Using OS tools (netstat etc.) to understand TCP protocol (connection establishment, connection termination, retransmission timer behaviour, congestion control behaviour)

Setting up a small IP network - configuring interfaces, IP addresses and routing protocols to set up a small IP network

Introduction to network simulator tools (NS-2/3, GNS3, Mininet, QualNet etc.) to study behaviour of MAC (IEEE 802.3, IEEE 802.11) and other protocols.

Books:

1. WR Stevens, UNIX Network Programming, 2nd Ed., 2015, Pearson
2. Kirch and Dawson, Linux Network Administrator's Guide, O'relly

References:

1. Online and weblink reference manuals for network simulators

Project I (using Software Engineering perspective) CO317

0-0-2 :2 Credits: 4 Hours Prerequisites:CO311

Abstract: Each student will carry out a project work individually under the guidance of a faculty member. The project shall consist of design/ development/ implementation work with a view to understand and apply Software Engineering principles to construct software that is reliable, reasonably easy to maintain. Students will gain hands-on experience on the design and development of a software system through application of software engineering knowledge and skills.

Course Outcomes:

After the course, a student will be able to:

1. Earn practical knowledge of developing a software system.
2. Apply the software engineering concepts in a real-life software project.
3. Earn an exposure to technology framework and tools for software development.
4. Develop test suites based on software testing principles, test and debug software modules in Java and C++ etc.
5. Use of appropriate CASE tools and other tools such as configuration management tools, program analysis tools in the software life cycle
6. Prepare relevant project documents.

Course Contents:

Design, development and deployment of software system over multiple iterations; Application of software engineering concepts and knowledge such as requirements engineering, software architecture, design, development, testing and validation; Usage of technology frameworks and tools for software development; Exposure to real world software development environment under constraints of uncertain requirements and deadlines.

References:

1. Rajib Mall, Fundamentals of Software Engineering, Prentice Hall India Learning Private Limited (Fifth Edition, 2018)
2. R.S. Pressman, Software Engineering: A Practitioner's Approach, McGraw Hill, 2014
3. M. Shooman, Software Engineering, McGraw Hill, 1983
4. Roy Osherove, The Art of Unit Testing, Second Edition, Manning Publications; 2nd edition (December 7, 2013)
5. Steve McConnell, Code Complete: A Practical Handbook of Software Construction, Microsoft Press; 2nd edition (June 19, 2004)
6. Emma Jane Hogbin West, Git for Teams: A User-Centered Approach to Creating Efficient Workflows in Git, O'Reilly Media; 1 edition (September 12, 2015)

Artificial Intelligence	CO401
3-0-0: 3 Credits : 3 Hours	Prerequisites:CO210

Abstract:

This course is aimed at providing an overview of the various aspects and behaviours of intelligent systems. It covers the various techniques that have been devised to emulate intelligent behaviour in artefacts. The course discusses ideas related to perception, reasoning, learning, acting and communicating in a complex environment.

Course Outcomes:

1. Getting students to appreciate the foundations of Artificial Intelligence and its future trends
2. Understanding of the basic elements constituting problems in an intelligent system and various approaches of solving them
3. Understanding the notions of uncertainty and decision making in real world problems
4. Introduction to learning mechanisms by which an intelligent system can improve its behaviour

Course contents:

Introduction to AI, background, related field

Uninformed search strategies: Breadth First Search, Depth First Search, Depth Limited Search, Iterative Deepening Depth First Search, Bidirectional Search and comparisons, Hill Climbing, Simulated Annealing

Informed Search Strategies: Heuristic Functions Significance of heuristic Functions, Desirable Properties, Design of Heuristic Functions Greedy Best First Search, A*

Search, Constraint Satisfaction Problems (CSP): Backtracking Search

Adversarial Search for Game Playing: Minimax Algorithm, Alpha-Beta Pruning, Making Real-Time Decisions

Knowledge Representation and Reasoning: Propositional Logic and Inference, Resolution principle, First Order Logic and Inference

Planning: Planning with State Space Search, Planning Graphs, STRIPS, Planning in the Real world, POP
Reasoning Under Uncertainties: Bayesian Networks, Exact and Approximate Inference, Expressing vagueness and ignorance

Markov and Sequential Decision Problems: Definition, Optimality in SDP, Algorithms for optimal Policies (Value Iteration, Policy Iteration)

Overview of Machine Learning: Goals and forms of Learning, Performance Assessment, Decision Theory, Inductive Learning, Concept Learning, Statistical Learning Methods, Introduction to Neural Networks, Genetic Algorithms, Reinforcement Learning.

Selected applications

Text Books

1. Rusell, S., and Norvig, P., Artificial Intelligence: A Modern Approach, 4th Ed, Pearson, 2020.

Reference Books

1. Koller, D., and Freidman N., Probabilistic Graphical Models, MIT Press, 2009
2. Mitchell, T., Machine Learning, Tata-McGraw-Hill, 1997.
3. Bishop, C. M., Pattern Recognition and Machine Learning, 2nd edition, Springer, 2006.

Project II	CO402
0-0-4: 4 Credits: 08 Hours	Prerequisites: CO311

The students will carry out project works in groups of 2 or 3 students each under the guidance of a faculty member. The project shall consist of research/ design/ development/ implementation work.

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

1. Conduct a background study of the topic of interest
2. Practically apply the basic computer science and engineering knowledge acquired 3. Communicate and present their work orally and in written.
4. Conceive, design, and implement projects using the inherent tools of engineering.
5. Organize, plan, and distribute the project-related works amongst the members of a group in a coherent manner.

Project III	CO403
0-0-8: 08 Credits: 16 Hours	Prerequisites: CO402

The students will carry out project works in groups of 2 or 3 students each under the guidance of a faculty member. The project shall consist of research/ design/ development/ implementation work. It may also be a continuation of the Project II work.

Course Outcomes:

At the end of the project the student will be able to:

1. Conduct a background study of the topic of interest
2. Practically apply the basic computer science and engineering knowledge acquired 3. Communicate and present their work orally and in written.
4. Conceive, design, and implement projects using the inherent tools of engineering.
5. Organize, plan, and distribute the project-related works amongst the members of a group in a coherent manner.

Controller of Examinations
Tezpur University
Tezpur, Assam

CS606	Computer Architecture and Parallel Processing	3-0-0	3

COURSE CONTENT

Definitions of Computer Architecture - Abstract Architecture & Concrete Architecture.
Concepts in Parallel Processing - Available Parallelism and Utilized Parallelism.
Parallel Programming Models – PRAM, Shared Variable, Message Passing, Data Parallel.
Classification of Computer Architectures – Flynn's Classification – Classification of Parallel Architectures.
Instruction Level Parallel (ILP) Processors – Pipelined, VLIW, Super Scalar Processors – Instruction Dependencies, their Effect on Performance and Techniques to overcome them.
Basic Concepts and Techniques in Vector, Systolic and Dataflow architectures.
Multiprocessor Architectures – Synchronization and Cache Coherence Issues.
Multicomputer Architectures – Interconnection Networks, Routing and Data Communication Algorithms.

Books/References:

1. D. Sima, T. Fountain, P. Kacsuk, Advanced Computer Architectures – A Design Space Approach, Addision-Wesley.
2. K. Huang, F. A. Briggs, Computer Architecture and Parallel Processing, McGraw Hill.
3. V. Kumar et el. Parallel Computing, Klewer Publishers.

CS 543	Advanced Programming Lab I	0-1-2	3

PREREQUISITES

Undergraduate courses in procedural programming and object oriented programming

COURSE OBJECTIVES

- Be competent with use of basic constructs provided by high-level programming languages
- Be able to use computational thinking to design solutions
- Be competent with use of computational approaches to solve problems in science and engineering
- Develop proficiency in implementation of Algorithmic techniques using appropriate data structures
- Be proficient in using advanced data structures
- Develop competency in implementation of advanced algorithms

COURSE CONTENT

This course will involve the students extensively in writing programs for assignments in contemporary programming languages. Topics include:

- Matrix operations like inverse, rank, linear equations, Eigen values and vectors
- Implementing algorithms using recursions, divide-and-conquer, greedy, dynamic programming concepts
- Linear programming using Cplex/LINGO.
- Implement some commonly used graph algorithms
- Implement network flow algorithms/matching
- Implement advanced data structures like B-Tree, dynamic trees, skip lists, suffix tree, red black tree, tries, R-tree, geometric data structures (KD-tree, interval tree, Quad tree, range tree etc.)

COURSE OUTCOMES

After completion of course, students would be able to:

- Master in use of procedural programming language C and object oriented language C++ to implement basic algorithms and use of data structures
- Master in development and implementation of algorithms for different problems in Computer Science
- Master in implementation of graph algorithms for some well known problems
- Master in solving some linear programming problems using Cplex/LINGO
- Master in implementation of advanced data structures for solving problems related to cryptography, networks, machine learning, text processing, internet programming, compiler construction etc.

Books/References:

Text Books:

1. Introduction to Scientific Computation and Programming, Daniel T. Kaplan
2. T. Cormen, C. Leiserson, R. Rivest, and C. Stein, Introduction to Algorithms, 2nd edition, 2001.
3. Fischer and Leblanc, Crafting a Compiler with C
4. Clean Code: A Handbook of Agile Software Craftsmanship by Robert C Martin
5. Programming Pearls by Jon Bently
6. Algorithms + Data Structures = Programs by Niklaus Wirth
7. Thinking in C++ by Bruce Eckel
8. Thinking in Java by Bruce Eckel
9. Effective Java by Joshua Bloch
10. Python in Practice. Mark Summerfield
11. Python Cookbook. David Beazley and Brian K. Jones

IT 510	Advanced Operating Systems	3-0-1	4

COURSE CONTENT

Multirocessor issues: Threads-usage, design issues and Implementation, synchronization primitives. Processor Management & Scheduling, Memory management.
Real-time operation System: Design issues and Implementation
Advance Programming Laboratory in UNIX: Drivers, scheduler, threads, IPC, file system.

Books/References:

1. Tanenbaum, Modern Operating Systems, PHI (EEE)
2. Milenkovic, Operating Systems: Concepts and Design, McGraw Hill.
3. Sillberschatz et. al, Operating Systems.
4. W.R. Steveans, Advanced Progamming in the UNIX Environment, Addison Wesley.
5. M.J. Bach, The Design of the UNIX Operation System, PHI(EEE).
6. Singhal and Shivaratri, Advanced Concepts in Operating Systems, TMH

CS 634	Selected Topics in Computer Networks	3-0-1	4

COURSE CONTENT

General congestion Control & Queuing, TCP Congestion Control, Random Early detection(RED) Gateways for Congestion Avoidance, Tuning RED for Web Traffic, Core Stateless Fair Queuing, The war between Mice & Elephants, Promoting the Use of End-to-End Congestion Control in the Internet, A study of Active Queue Management for Congestion Control, Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management, Congestion Control for High bandwidth-Delay Product Networks, Improving the Performance of Reliable Transport protocols in Mobile Computing Environments, End-to-End Bandwidth Estimation in TCP to Improve Wireless Link Utilization, Discriminating Congestion Losses from Wireless Losses using Inter-Arrival Times at the Receiver.

SCTP – Multi-homing, Multi-streaming features, Application in High Availability, Mobility, Multimedia and Web Data Transport, Congestion Control and Security Issues.

Active Networks, P2P networks and Overlay Networks design issues, Network Performance and modeling, Wireless networks, ad-hoc networks and sensor networks.

Security issues in different networks.

Books/References:

1. Computer Networks, A Systems Approach, Third Edition, L. Peterson and B.S. Davie, Morgam Kaufmann, 2003.
2. Computer Networking, A Top-Down Approach Featuring the Internet, Second Edition, J.F. Kurose and K.W. Ross, Adition-Wesely, 2002.
3. Computer Networks, Fourth Edition, A. Tanenbaum, Prentice-Hall, 2002.

CS 545	Seminar	0-2-0	2

PREREQUISITES

Major core courses of M.Tech IT program

COURSE OBJECTIVES

- to create an environment to engage students in delivering and listening to interesting talks that promotes discussion
- to provide students with opportunity to learn new concepts and skills acquired in core courses and further extend these ideas to solve research/industry related problems
- know how to read research papers critically and efficiently
- to learn fundamental principles, generalizations and important theories of Computer Science
- to enable students to find their own field of interest in academia, industry or entrepreneurship
- to help students develop their own learning and teaching styles and communication skills

COURSE CONTENT:

Student presentations:

- Each student will present one paper during the term

Class evaluations:

- Each week each student is asked to write a short evaluation of one of the papers being presented

Class Discussion:

- Discuss the papers – expose the flaws, analyse the writing, what was the impact?

COURSE OUTCOMES

Because the curriculum is about individuals, there are no specific course levels learning outcomes. Nevertheless, after completion of the course, students would be able to:

- Explain factual knowledge (terminology, classifications, methods, trends). of current areas of research.
- State and explain some fundamental principles, generalizations, or theories the student has learned in this course.
- To apply gained knowledge in thinking, problem solving, or decisions making process.
- To achieve specific skills, competencies, and points of view needed by computing professionals.
- to judge the value of different contributions
- to identify promising new directions

Books/References:

A list of works will be posted by mentors/teachers at the start of the course. The students also have the option of choosing works according to his/her own areas of interest.

CS 531	Object Oriented Programming and Design	3-1-1	5

COURSE CONTENT

Part I : Object Oriented Programming

Structured Programming and Object Oriented Programming paradigms. Key Concepts:

Data Abstraction: Class, object, constructors, destructors, memory allocations for objects, member functions, friend functions, templates.

Inheritance: Single & multiple inheritance, virtual base class.

Polymorphism: Compile time polymorphism: operator overloading, function overloading, static binding.

Run-time polymorphism: Virtual function, pure virtual function, abstract class, dynamic binding.

Exception handling.

Part - II Object Oriented Design

Object Oriented Design Approaches: Object Model, Dynamic Model, and Functional Model. (Objet Diagram, State Diagram, and DFD).

Phases of Object Oriented Development: Object Analysis, System Design, Object Design.

Books/References:

1. Herbert Schild : The Complete Reference to C++, Osborne McGrawHill.
2. Bjarne Stroustrup: The C++ Programming Language, Addison Wesley
3. Rambaugh et al. : Object Oriented Modeling and Design, PHI(EEE).
4. Grady Booch: Object Oriented Analysis and Design, Pearson Education.

CS 601	Design and Analysis of Algorithms	3-0-0	3

COURSE CONTENTS

Review of basic data structures such as stack, queue, linked list, trees and graphs. Concepts in algorithm analysis, Asymptotic complexity.

Domain independent algorithm design techniques such as divide and conquer, greedy method, dynamic programming, back tracking, branch and bound. Basic ideas about neural network, genetic algorithms and simulated annealing.

Example algorithms for sets, graphs, text processing, internal and external sorting, height balanced trees, B-trees, hashing, dynamic storage allocation, garbage collection.

Lower bound theory and NP-hard problems.

Books/References:

1. Corman et al., Introduction to Algorithms, McGrawHill.

Aho A, Hopcroft J., Ullman J., The Design and Analysis of Algorithms, Addison- Wesley.

CS 508	Database Management Systems	2-1-2	5

COURSE CONTENT

Overview:Concept of database, data independence, redundancy Control; Database architecture - ANSI model.
Modeling of real world situation: Entity-relationship model; Data models: Network, Hierarchical, Ralational.
Relational data model: DDL, DML: relational algebra and calculus; functional dependencies, normal forms, decomposition, integrity rules; Query languages for relational systems: SQL, QBE, query optimization, embedded SQL.
Database transactions, concurrency control, recovery and security issues in databases.
Brief treatment of: Client-server models, distributed databases, object-oriented databases, deductive databases, multimedia databases, active databases.

Books/References:

1. Silberschatz and Korth, Database system concepts, McGraw Hill.

2. Elmasri and Navathe, Fundamentals of database systems; Narosa Publishing Co.

CS 505	Software Engineering	3-0-1	4

COURSE CONTENTS

Introduction to software engineering, concept of a software project, size factor, quality and productivity factor, different phase of a software development life cycle, managerial issues.

Software project planning: Problem definition, development of a solution strategy, development process planning, software development models and their comparative study; Organizational structure planning, project formats and team structures; Planning for quality assurance and configuration management; Planning for verification and validation.

Software economics: Cost estimation and evaluation techniques, cost estimation based on COCOMO model and Raleigh model.

Software requirements analysis and specifications techniques- their notations & languages . Software design: Concept of fundamental design; Design approaches- top-down & bottom-up, structured, object-based & object oriented design; Design specification and notations.

Software implementation: Structured coding techniques, coding styles, and standards; Guidelines for coding and documentation.

Software verification and validation: Theoretical foundation, black box and white box approaches; Integration and system testing.

Software reliability: Definition and concept of reliability, software faults, errors, repair and availability, reliability and availability models.

Case studies.

Books/References:

1. Pressman, R.S., Software Engineering: A Practitioner's Approach, McGraw Hill.
2. Shooman, M, Software Engineering, McGraw Hill.
3. Fairley, R.E., Software Engineering Concepts, McGraw Hill.

CS 507	Computer Networks	3-0-1	4

COURSE CONTENT

Review of Computer Network Architecture and the Subnet layers.
Data Transport: Connection management, Quality of Service, TCP/IP Protocol, ATM.
Session Management: Session establishment and maintenance, Dialogue management, Recovery. End-to-end Data: Presentation formatting issues and methods: XDR, ASN.1, NDR; Data Compression, Lossless Compression Algorithms- Run length encoding, DPCM, Dictionary- based methods, Image compression- JPEG, Video compression- MPEG; Security and authentication techniques, Encryption algorithms.
Applications: E-mail, Remote login, File transfer, Network file system, Network management. UNIX network programming with TCP/IP; Network File System, Novell Netware, and Windows
NT installation, configuration and use.

Books/References:

1. Tanenbaum A.S., Computer Network, 3e, PHI (EEE).
2. Stalling W, Data and Computer Communication, 5e, PHI (EEE).
3. Peterson L L, Davie B S, Computer Networks: A Systems Approach, Morgan Kaufmann Publishers Inc.
4. Stevens, UNIX Network Programming, PHI (EEE).
5. Comer, TCP/IP Programming Vol.- I, II, III, PHI(EEE).

Course Name: Mathematical Foundation for Computer Science Course

Code: CS541

Credits: 4 (L-3 T-1 P-0)

PREREQUISITES

Undergraduate course in Discrete Mathematics/Discrete Structures

COURSE OBJECTIVES

- Have the basic statistical methodology of data analysis including; graphs, descriptive statistics
- Understand random variables and its distributions
- Use statistical tests in testing hypotheses on data
- To provide the foundations of probabilistic and statistical analysis that can be helpful in modeling applications like gene expression, stock market prediction, pattern recognition, computer networks etc.
- To provide concepts of Linear Algebra that are mandatory in the areas including graphics, image processing, cryptography, machine learning, computer vision, optimization, graph algorithms, quantum computation, computational biology, information retrieval and web search

COURSE CONTENT

Review of Discrete Mathematics:logic, set theory, functions and relations, , proof techniques, algorithms and integers, counting, generating functions, graphs, combinatorics

Basic probability: Basic probability: basic idea, expectation, probability calculus, Bayes' theorem, conditional probability, Data summaries and descriptive statistics, central tendency, variance, covariance, correlation, Probability distribution functions: uniform, normal, binomial, Poisson, chi-square, student's t-distribution, central limit theorem, Stochastic processes, Markov chains

Mathematical Statistics: Sampling, measurement, error, random number generation, Hypothesis testing, A/B testing, confidence intervals, p-values, regression and analysis of variance

Linear Algebra: Basic properties of matrix and vectors: scalar multiplication, linear transformation, transpose, conjugate, rank, determinant; Inner and outer products, matrix multiplication rule and various algorithms, matrix inverse; Special matrices: square matrix, identity matrix, triangular matrix, idea about sparse and dense matrix, unit vectors, symmetric matrix, Hermitian, skew-Hermitian and unitary matrices
Applications of Algebra:Linear Systems. Matrices. Gauss elimination. Echelon form. Matrix multiplication. Elementary matrices. Inverse matrix.; Vector space, basis, span, orthogonality, orthonormality, linear least squareEigen values, eigenvectors, characteristic polynomial, diagonalization, singular value decomposition

Dealing with intractability: Recent trends in the use of statistics and algebra in various fields of computer science including bioinformatics, machine learning, computer vision, pattern recognition etc.

COURSE OUTCOMES

After completion of course, students would be able to:

1. read a real world problem,
2. realize the uncertainty that is involved in a situation described,
3. select a suitable probability model,
4. estimate and test its parameters on the basis of real data,
5. compute probabilities of interesting events and other vital characteristics, and make appropriate conclusions and forecasts
6. apply linear algebra concepts in two-dimensional graphics transformations, face morphing, face detection, image transformations (e.g., blurring and edge detection, image perspective removal), classification of tumors as malignant or benign, integer factorization, errorcorrecting codes, and secret-sharing.

Books/References:

Text Books:

1. John Vince, Foundation Mathematics for Computer Science, Springer.
2. K. Trivedi. Probability and Statistics with Reliability, Queuing, and Computer Science Applications. Wiley.
3. M. Mitzenmacher and E. Upfal. Probability and Computing: Randomized Algorithms and Probabilistic Analysis.
4. Alan Tucker, Applied Combinatorics, Wiley
5. Sheldon Axler. Linear Algebra Done Right, Springer
6. Gilbert Strang. Linear Algebra and Its Applications. Cengage Learning
7. David A. Forsyth, Probability and Statistics for Computer Science, Springer
8. Sheldon M. Ross, Probability Models for Computer Science, Academic Press

Course Name: Advanced Algorithms and Data Structure Course

Code :CS542

Credits: 3 (L-2 T-1 P-0)

PREREQUISITES

Undergraduate course each in algorithms and data structures

COURSE OBJECTIVES

- Develop 'students' algorithmic thinking and their ability to analyse the efficiency of algorithms;
- Enable students to find different approaches for dealing with challenging computational problems;
- Provide insight into cutting-edge research-led teaching in modern subfields of algorithms theory.

COURSE CONTENT

Algorithm Review:Basic algorithms: Asymptotic notation, amortized analysis, average case analysis

Review of Basic Data Structures: arrays, heap, lists, binary search trees, hashing, graphs

AlgorithmicTechniques: recursion, divide-and-conquer, greedy, dynamic programming,

Advanced Data structures:Disjoint sets / union-find, Fibonacci heaps, splay trees, dynamic trees, Red-Black trees, Skip lists, B-tree, dictionaries, geometric data structures

Advanced Algorithms:Fast Fourier transform, network flow, Linear programming, Matching algorithms, String matching, Number theoretic algorithms + RSA, geometric algorithms

Dealing with intractability:NP-Completeness, Approximation algorithms, LP based approximations, randomized algorithm

COURSE OUTCOMES

After completion of course, students would be able to:

- Appreciate what solution to a computational problem efficient or inefficient
- Compare between different data structures. Pick an appropriate data structure for a design situation.
- Analyse the efficiency of advanced algorithmic techniques e.g., approximation(randomized and LP-based rounding etc), randomization algorithms (expected running time, probability of error).
- Identify and apply design principles for the design of advanced efficient algorithms
- Understand some pieces of current research on algorithms.

Controller of Examinations
Tezpur University
Tezpur, Assam

Books/References:

Text Books:

1. Rajeev Motwani and Prabhakar Raghavan, Randomized Algorithms, Cambridge University Press, 2000.
2. Michael Mitzenmacher and Eli Upfal, Probability and Computing, Cambridge University Press, 2005.
3. David Williamson and David Shmoys, The Design of Approximation Algorithms, Cambridge University Press, 2011.
4. Jon Kleinberg and Eva Tardos, Algorithm Design, Addison-Wesley, 2006.
5. T. Cormen, C. Leiserson, R. Rivest, and C. Stein, Introduction to Algorithms, 2nd edition, 2001.

Course Name: Advanced Programming Lab II Course Code: CS544

Credits: 3 (L-0 T-1 P-2)

PREREQUISITES

Undergraduate courses in procedural programming and object oriented programming

COURSE OBJECTIVES

- Be competent with Linux shell programming
- Be competent in processing different types of textual data using
  Python/Perl/Java/Lex/Yacc
- Be competent with writing simple MATLAB/Octave programs for performing numerical calculations
- Be competent to use ML approaches to solve problems in Bioinformatics, Computer Visions,
  Pattern Recognition, Image Processing, Cyber security, knowledge discovery, NLP etc
- Be competent in using language and tools to solve problems of Graphics, Computer Vision and use of simulation tools
- Be proficient in high quality document preparation using packages including LaTex, GNU plot etc

COURSE CONTENT

This course will involve the students extensively in writing programs for assignments involve programming in contemporary programming languages. Topics include:

- Unix/Linux shell programming,
- Documentation using LaTex
- Text processing using Python/Perl/Java Script
- Text parsing using Lex/Yacc
- Use of ML approaches for solving computational problems
- Writing codes for image processing, graphics, computer vision related problems C/C++/Java/MatLab/Octave

COURSE OUTCOMES

After completion of course, students would be able to:

1. Master an understanding of scripting and the contributions of scripting languages.
2. Master an understanding of Python especially the object-oriented concepts,
3. Master an understanding of the built-in objects of Python,
4. Master in using ML approaches in computational problems
5. Master in preparation of high quality documents using LaTex and different graphics packages
6. Master in using simulation tools

Books/References:

Reference Books:

1. LaTeX Beginner's Guide, Stefan Kottwitz
2. Latex Document Preparation System Users, Leslie Lamport
3. Introduction to Computation and Programming Using Python: With Application to Understanding Data. MIT Press, Guttag, John.
4. Crafting a Compiler with C, Fischer and Leblanc
5. Programming Pearls, Jon Bently
6. Flex & Bison, John R. Levine
7. A Guide To Matlab: For Beginners And Experienced Users, Hunt, Lipsman and Rosenberg
8. Matlab for Machine Learning: Functions, algorithm and use cases, Giuseppe Ciaburro
9. Thinking in C++, Bruce Eckel
10. Thinking in Java, Bruce Eckel
11. The C programming Language, K&R
12. Advanced Programming in Unix Environment, R. Stevens
13. Eloquent JavaScript A Modern Introduction to Programming,MarijnHaverbeke
14. Unix Shell Programming, Sams Publishing,Stephen G. Kochan and Patrick Wood
15. Programming Perl: Unmatched power for text processing and scripting, Tom Christiansen, Brian DFoy, Larry Wall and Jon Orwant
16. Perl Best Practices,O'Reilly, Damian Conway
17. John W. Eaton, David Bateman, Soren Itauberg, Rik Wehbring (2019). GNU Octave version 5.1.0 manual: a high-level interactive language for numerical computations. https://www.gnu.org/software/octave/doc/v5.1.0/

Course Name: Term Project I Course

Code : CS640

Credit : L-0:T-0:P-8: 8

PREREQUISITE: NIL

COURSE OUTCOMES

- Demonstrate sound fundamentals in a chosen area of computing
- Identify and formulate a problem of research interest in the chosen area of computing
- Analyze the computing problem and propose solutions
- Apply the emerging technologies like – Blockchain, IoT, Robotics, ML, AI, Datamining, Big Data Analytics in solving some challenging problem in chosen area
- Effectively communicate the work at all stages of the project

COURSE CONTENT

The student is expected to carry out supervised research in this course. An intensive literature in thechosen area, should result in sound knowledge in the area and result in the identification of a suitable research problem, and its formulation and analysis. Study of relevant supplementary literature, such mastering useful programming languages and tools for the problem, are also expected at this stage of the project. The student is expected to present three reports at different evaluation points duringthe semester, with clearly defined achievements and plans for further steps.

References

Relevant literature and software tools for the chosen problem

Controller of Examinations
Tezpur University
Tezpur, Assam

Course Name: Term Project II Course

Code : CS641

Credit: L-0: T-0:P-16: 16

PREREQUISITE:Term Project I

COURSE OUTCOMES

- Identify a suitable problem to be solved computationally
- Reflectively analyze proposed solutions to the identified computing problem
- Design and develop solutions to the problem and analyze results
- Prepare a thesis and defend the thesis on the work done
- Augment the knowledge base in the chosen area of computing, adhering to ethical practices at every stage

COURSE CONTENTS

The studentsare expected to demonstrate the core competency in the development of enhancements to the knowledge base in the area of interest in computing. The secondary competencies include the management of time bound projects involving research, analysis of problem complexities, design and development of effective solutions and communication of the project's progress, adhering to ethical practices at every stage. This stage of the project evaluates the state of maturity of these competencies. The students are expected to present two reports at intermediate stages, as well as prepare and defend a thesis on their research work.

The students will usually continue the project work in Term Project I (CS 641) or optionally can take a new research-oriented project in consultation with the assigned project supervisor.

References

Relevant literature and software tools for the

***