B.C.S.E THIRD YEAR FIRST SEMESTER

Course code	CSE/PC/B/T/311
Category	Professional Core
Course title	Computer Graphics
Scheme and Credits	L–T–P: 3-0-0; Credits: 3.0; Semester – I
Pre-requisites (if any)	

Syllabus:

Introduction:
Brief discussion on historical perspective; graphics primitives such as points, lines, polygons, etc.; representation of pictures using primitives; storage & retrieval of pictures; introduction to graphics display devices; calligraphic/ vector graphics versus raster graphics; bit plane; colour look-up table; introduction to graphic input devices – track ball, mouse, digitizing tablet, light pen etc. [2L]

Rasterization techniques: [2L]
Line – DDA; Bresenham's generalized integer version; Mid-point rasterization. Circle – Bresenham's algorithm; Mid-Point algorithm - 1st order difference & 2nd order difference methods. [3L]
Ellipse – Mid-Point algorithm - 1st order difference method, brief discussion on 2nd order difference method [1L]

2D Scan conversion & polygon filling:
Active-Edge-List (y-bucket) scan conversion of lines & polygons; [1L]
Edge –fill , Fence –fill, & Edge –flag polygon filling algorithms; simple Seed –fill & Scan –line seed –fill algorithms. [2L]

2D Geometric transformations:
Introduction to position vector; representation of 2D objects as matrices; transformation matrices for scaling, shear, rotation, reflection [2L]
homogeneous coordinates; representing translation as a transformation matrix; composite transformation matrix for arbitrary transformation; invariance of origin under transformation; [2L]
invariance of parallelism under transformation; transformation of intersecting lines; area of transformed polygons; 2D view-port & viewing window. [2L]

2D Clipping:
Clipping against regular window – Explicit line clipping; [1L]
Sutherland & Cohen line clipping, [1L]
Mid-point subdivision line clipping; [1L]
Clipping against arbitrary convex window – Cyrus Beck clipping algorithm, [1L]
Liang Barsky clipping algorithm; [1L]
Sutherland & Hodgemann polygon clipping. [1L]

3D Graphics:
Introduction to right handed coordinate system for 3D representation; matrix representation of 3D object; scaling, shear & translation transformation; [1L]

rotation about principal coordinate axes & about arbitrary line; composite transformation for arbitrary 3D transformation. [2L]

Projection:

Introducing the idea of projecting 3D object on to 2D plane; broad classification – parallel & perspective projection; different types of parallel projection & examples of each; [1L]

formal definition of 3D to 2D projection and derivation of projection matrix; 1-point, 2-point & 3-point perspective projection; formal derivation of vanishing point(s) and physical implication of the same. [2L]

Curves:

Introduction to curve fitting; piece-wise approximation using known curves; approximation using different functions – polynomial, exponential, trigonometric etc.; [1L]

Introduction to blending function; detailed illustration by creating a hypothetical polynomial blending function; [1L]

General spline; cubic spline; B- spline; Hermite curve; boundary & continuity conditions for these curves; [2L]

Bezier curve; 1st & 2nd order continuity conditions for joining Bezier curves; splitting Bezier curve; [2L]

Hidden line removal:

Introduction; simple z-buffer algorithm; scan line z-buffer algorithm; floating horizon algorithm. [2L]

Basic interaction handling:

Different classes of devices – locator, pick, valuator etc.; input & output handling in a window system. [1L]

Illumination & shading:

Introduction; basic illumination models; ambient, diffused and specular reflection light models; simple flat/ faceted shading; Gourad shading; Phong shading; simple ray tracing algorithm, Color models [2L]

Books:

1. Procedural Elements for Computer Graphics by David F. Rogers, TMH publication.
2. Mathematical Elements for Computer Graphics by David F. Rogers and J. A. Adams, TMH publication.
3. Computer Graphics, principles & practices by J.D. Foley, A. van Dam, S. K. Feiner and J. F. Huges, Addison Wesley.
4. Computer Graphics, C version, by D. Hearn and M. P. Baker, Pearson Education.
5. Computer Graphics, a programming approach, by S. Harrington, TMH publication.
6. Computer Graphics by A.N. Sinha and A. D. Udai, TMH publication

Course Outcomes (COs):

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

CO1: Understand the graphical primitives, describe the concept of rasterzation and develop line, circle and ellipse drawing algorithms, filling algorithms.
CO2: Understand 2D Geometric Transformations, including clipping algorithms.
CO3: Understand the concept of 3D graphics, including 3D transformation and projection
CO4: Develop curve fitting algorithms, and describe the concepts of hidden line removal, illumination and shading.

Course code	CSE/PC/B/T/312
Category	Professional Core
Course title	Systems Programming
Scheme and Credits	L–T–P: 3-0-0; Credits: 3.0; Semester – I
Pre-requisites (if any)	

Syllabus:

1. Introduction to Systems Programming [**2L**]
2. Assembler - Functions of an assembler, features (with respect to machine dependence) of an assembler, design of assembler – two pass, one pass, the concept of overlay [**6L**]
3. Assembler Design: A case study – Overview of 16 / 32 bit architecture and assembly language programming features, Functionalities and design of assembler for such specifications [**6L**]
4. Macro processor – Functions of macro processor, features of macro processor, design of macro processor [**4L**]
5. Loaders and Linkers – functions of loader, absolute loader, bootstrap loader, Machine dependent and machine independent features of loader, Relocation, Linking, concept and design of relative/relocating loader, linking loader, linkage editor, dynamic linking and dynamic loading [**8L**]
6. Text editor – types of editors, types of files, features, examples [**2L**]
7. Cross assembler – requirements, features, example [**2L**]
8. Debug – functions of a debugger, hardware support for debugging, example debuggers [**2L**]
9. IDE - Assembly Language Programming Paradigm, Components - Environment, Assembler, Source Editor, Emulator etc. [2L]
10. Device drivers – concepts, design and development [**6L**]

Suggested Readings:

1. L. Beck, System Software: An Introduction to Systems Programming –

2. M. Joseph, System Software –, Laxmi Publications

3. D. M. Dhamdhere, Systems Programming and Operating Systems, TMH4.

4. J. Corbet, A. Rubini, G. Kroah-Hartman, Linux Device Drivers –, O' Reilly Media

Course Outcomes:

CO1: Able to comprehend the basics of system programs like editors, compiler, assembler, linker, loader, interpreter and debugger.

CO2: Able to describe the various concepts and functionality of designing assemblers and macro processor.

CO3: Able to analyze how linker and loader create an executable program from an object module created by assembler.

CO4: Able to interpret the various concept of Text editors, cross assembler etc.

CO5: Able to design programs to develop assembler, loader, linker, text editor etc. in a systematic way

CO6: Able to comprehend the fundamental concepts of device drivers with case study

Course code	CSE/PC/B/T/313
Category	Professional Core
Course title	Operating System
Scheme and Credits	L–T–P: 3-0-0; Credits: 3.0; Semester – I
Pre-requisites (if any)	

Syllabus:

1. Introduction to Operating Systems **[1L]**
2. Concept of batch-processing, multi-programming, time sharing, real time operations **[2L]**
3. Process Management: Concept of process, state diagram, process control block; scheduling of processes – criteria, types of scheduling, non-preemptive and preemptive scheduling algorithms like: FCFS, Shortest Job First/Next (SJF/N),Shortest Remaining Time Next (SRTN), Round Robin (RR), Highest Response ratio Next (HRN), Priority based scheduling, different Multilevel queue scheduling etc. **[5L]**
4. Threads – concept, process vs thread, kernel and user threads, multithreading models **[2L]**
5. Inter-process Communication (IPC) – Shared memory, message, FIFO, concept of semaphore, critical region, monitor **[2L]**
6. Process Synchronization: concepts, race condition, critical section problem and its solutions; synchronization tools- semaphore, monitor etc., discussion of synchronization problems like producer-consumer, readers-writers, dining philosophers, sleeping-barber etc. Deadlock conditions, resource allocation graph, prevention techniques, avoidance technique – Banker's algorithm and related algorithms **[6L]**
7. Memory management: Address space and address translation; static partitioning, dynamic partitioning, different types of fragmentation, paging, segmentation, swapping, virtual memory, demand paging, page size, page table, page replacement algorithms – FIFO, LRU, Optimal page replacement, Variants of LRU, etc; thrashing, working set strategy **[6L]**
8. File Management: File and operations on it, file organization and access; file allocation; directory structures, file sharing, file protection **[4L]**
9. Device management: Magnetic disks, disk scheduling- criteria, algorithms – FCFS, SSTF, SCAN, C-SCAN, LOOK, etc, disk management – formatting, boot block, disk free space management techniques, concept of RAID etc **[3L]**
10. Protection and Security: Concepts of domain, Access matrix and its implementation, access control, Security of systems- concepts, threats- Trojan horse, virus, worms etc, introduction to cryptography as security tool, user authentication **[5L]**
11. Case Studies including micro-kernel **[4L]**

Books:

1. Operating Systems Concepts – A. Silberschatz, P. Galvin and G. Gagne. Wiley India

2. Operating Systems Concepts - Gary Nutt, N. Chaki and S. Neogy, Pearson Education
3. Operating Systems – W. Stallings, Pearson Education
4. Operating Systems: A Concept-based Approach – D. M. Dhamdhere, Tata McGraw-Hill

Course Outcomes (COs):

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

CO1: Comprehend the concept and functionalities of OS.
CO2: Develop process management algorithms for scheduling, synchronization, deadlock and demonstrate competence related to scheduling, synchronization, multithreading.
CO3: Analyze various memory management techniques along with demonstration of dealing with the challenges of main memory, virtual memory.
CO4: Illustrate issues related to disk and file system management along with implementation techniques.
CO5: Identify protection and security issues and corresponding techniques.
CO6: Explain OS architecture (Linux/Unix/Mach OS/Windows).

Course code	CSE/PC/B/T/314
Category	Professional Core
Course title	Formal Languages and Automata Theory
Scheme and Credits	L–T–P: 3-0-0; Credits: 3.0; Semester – I
Pre-requisites (if any)	

Syllabus:

Finite Automata- **[7L]**
DFA, NFA, Recognition of a language by an automaton, Equivalence of DFA and NFA, Minimization of FA, Equivalence of FAs

Regular Languages- **[9L]**
Regular Sets and Languages, Pumping Lemma for Regular Languages, Closure Properties of Regular Sets, Kleen's Theorem

Context-free Languages and Push-Down Automata - **[10L]**
Non-regular languages, CFLs, Closure properties of CFLs, Grammars, Ambiguity, Push-Down Automata, Normal Forms, Pumping Lemma for CFL.

Turing Machines- **[8L]**
Introduction to Context Sensitive Languages and Grammars, Turing Machines and its variants, Universal TMs, Halting Problem, Recursive Functions and Sets, Recursively Enumerable Sets, Arithmetization of TMs.

Complexity Classes- **[6L]**
Space and Time Complexity, RAM programs and TMs, PTIME, NP, PSPACE etc., Polynomial reducibility.

Books:

1. Introduction to Automata Theory, Languages and Computation - J. E. Hopcroft and J. D. Ullman
2. Elements of the Theory of Computation - H. R. Lewis and C. H. Papadimitriou
3. An Introduction to Formal Languages and Automata – Peter Linz, Narosa
4. Introduction to the Theory of Computation – Michael Sipser, Thomson Press
5. Automata and Computability – Dexter C. Kozen, Springer

Course Outcomes (COs):

After studying the course, the students are expected to develop the following:

CO1: To be able to design Deterministic Finite Automata (DFA), convert Non Deterministic Finite Automata (NFA) to DFA, design Regular Expressions (REs), convert RE's to NFA and vice versa, and do state minimization of DFA.

CO2: To develop understanding of Pumping Lemma for Regular Languages (RLs).

CO3: To comprehend Decision properties and Closure Properties of RL's.

CO4: To be able to design Context Free Grammars (CFGs), understand relations between Parse trees and derivations of strings, comprehend Ambiguity of CFG's and LL(1) grammars, understand Chomsky Normal Form (CNF), design Push Down Automata (PDA), and understand equivalence between PDA and CFG's.

CO5: To develop understanding of Pumping Lemma for CFL's, Decision properties and Closure Properties of CFL's.

CO6: To develop understanding about Turing Machines (TMs), Universal Turing Machine (UTM), Church Turing Thesis, Recursively Enumerable Languages and Recursively Enumerable Sets, Chomsky Hierarchy, Halting Problem, Tractable and intractable problems.

Course code	CSE/PC/B/T/315
Category	Professional Core
Course title	Principles of Programming Languages
Scheme and Credits	L–T–P: 3-0-0; Credits: 3.0; Semester – I
Pre-requisites (if any)	

Syllabus:

Introduction: programming language definition, overview of programming paradigms. [2L]

Language design principles: design criteria, efficiency, regularity. [4L]

Syntax and Semantics: Context free grammar, BNF, syntax tree, parse tree, runtime environment, data types, type checking, weak typing, strong typing, static typing, parameter passing methods such as pass by value, pass by name, pass by result, pass by value-result, pass by reference. [6L]

Functional programming: distinctive features of functional programming languages, recursion, tail recursion, higher order functions, lazy evaluation, Introduction to functional programming using Java 8 Streams/Javascript/Scala/Python. [14L]

Lambda calculus: natural number operations, recursion, Boolean data types, objects, mapping of lambdas to modern programming languages. [8L]

Declarative programming: distinctive features of declarative programming, first order logic, Horn clauses, resolution unification, sequencing of control, negation, implementation issues, the language Prolog. [7L]

Object oriented paradigm: design and implementation issues in object-oriented languages, case study. [4L]

Suggested Readings:

1. Kenneth C. Louden, Programming Languages: Principles and Practice, 2003
2. D. A. Watt, Programming Languages and Paradigms, Prentice-Hall, 1990.
3. J. Lloyd, Foundations of Logic Programming, Springer Verlag, 1984.
4. M. Hennessey, The Semantics of Programming Languages, John Wiley, 1990.
5. Richard Warburton, Java 8 Lambdas, O'Reilly, 2014.
6. Raoul-Gabriel Urma, Mario Fusco, and Alan Mycroft, Java8 in Action, 2015.
7. W. F. Clocksin, C.S. Mellish , Programming in Prolog, Springer 2003.

Course Outcomes:
At the end of this course, each student will

CO1 : Be familiar with functional languages
CO2 : Be exposed to logic programming
CO3 : Be familiar with Lambda Calculus – universal model of computation
CO4 : Being able to map design concepts of Lambda Calculus to modern language features
CO5 : Be familiar with design issues of object oriented languages such as Ruby and Java
CO6 : Be able to apply proper programming paradigm or a mix of it depending on problem description

Course code	CSE/PC/B/T/316
Category	Professional Core
Course title	Computer Networks
Scheme and Credits	L–T–P: 3-0-0; Credits: 3.0; Semester – I
Pre-requisites (if any)	

Syllabus:

Introduction: Uses of Computer Networks, Types of Computer Networks, OSI Reference Model, TCP/IP model [2L]

Review of Physical Layer [2L]

Data Link Layer: Link Layer Addressing, ARP, Error detection and Correction Techniques, Flow and Error Control, Data Link Layer Protocols, MAC Protocols, Point to Point Protocol, Wired LANs– Ethernet, IEEE 802.3, Hub, Switches [10L]

Introduction to Wireless LANs, IEEE 802.11, MAC sublayer, Bluetooth [4L]

Network Layer: Introduction, Virtual Circuit and Datagram Networks, Performance metrics, IP Addressing, Subnetting, Routing Algorithms (Link State, Distance Vector, Hierarchical), Routing in the Internet (RIP, OSPF, BGP), Broadcast and Multicast Routing Algorithms, Routers, ICMP, Introduction toIPv6 [8L]

Transport Layer: Introduction to Transport Layer Services, Transport Layer Protocols, User Datagram Protocol (UDP), Transmission Control Protocol (TCP), SCTP, Introduction to Sockets and Socket Programming [8L]

Application Layer: SSH, Domain Name Space (DNS), Electronic Mail (SMTP, MIME, IMAP, POP3), File Transfer Protocol, [6L]

Suggested Readings:

1. Computer Networking: A Top-Down Approach Featuring the Internet, by James F. Kurose and Keith W. Ross, 5th Edition, Pearson Education, 2010

2. Data communication and Networking, by Behrouz A. Forouzan, 4th Edition, Tata McGraw-Hill, 2007

3. Computer Networks, by Andrew S. Tanenbaum, 4th Edition, Prentice Hall India, 2003

4. Computer Networks: A Systems Approach, by Larry L. Peterson and Peter S. Davie,4th Edition, Morgan Kauffman Publishers, 2007

5. Data and Computer Communication, by William Stallings, 9th Edition, Pearson Education, 2011

6. Unix Network Programming: Networking APIs: Sockets and XTI, (Volume 1) by W. Richard Stevens, 2nd Edition, Prentice Hall India, 1999

Course Outcomes (COs):

CO1 :Understand the layered architecture and explain the contemporary issues and importance of MAC sublayer of the Data Link Layer, network layer, transport layer and application layer of TCP-IP model, and how they can be used to assist in network design and implementation.

CO2 :Understand the protocols of MAC sublayer of the Data Link Layer and describe the IEEE standards for Ethernet (802.3) and wireless LAN (802.11).

CO3 :Understand internetworking principles and explain algorithms for multiple access, routing and different networking techniques and able to analyze the performance of these algorithms.

CO4 :Explain the protocols in Transport Layer and able to design network applications using these protocols.

CO5 :Explain the protocols in application layer and how they can be used to design popular internet applications.

Course code	CSE/PC/B/S/311
Category	Professional Core
Course title	Computer Graphics Lab
Scheme and Credits	L–T–P: 0-0-3; Credits: 1.5; Semester – I
Pre-requisites (if any)	

Syllabus:

Rasterization techniques:
Line – DDA; Bresenham's algorithm. Circle – Bresenham's algorithm; Mid-Point algorithm. Ellipse – Mid-Point algorithm

2D Scan conversion & polygon filling:
Edge –fill, Fence –fill, & Edge –flag polygon filling algorithms; simple Seed –fill & Scan –line seed –fill algorithms.

2D Geometric transformations:
representation of 2D objects as matrices; transformation matrices for scaling, shear, rotation, reflection homogeneous coordinates; composite transformation matrix for arbitrary transformation; invariance of origin under transformation;

2D Clipping:
Sutherland & Cohen line clipping, Mid-point subdivision line clipping; Cyrus Beck clipping algorithm, Liang Barsky clipping algorithm; Sutherland & Hodgemann polygon clipping.

3D Graphics:
Demonstration of right-handed coordinate system for 3D representation; scaling, shear & translation transformation; rotation about principal coordinate axes & about arbitrary line; composite transformation for arbitrary 3D transformation.

Projection:
Demonstration of the idea of projecting 3D object on to 2D plane; parallel & perspective projection; different types of parallel projection;

Curves:
General spline; cubic spline; B- spline; Bezier curve; splitting Bezier curve;

Course Outcomes (COs):
At the end of the course, a student will be able to:

CO1: Implement the Line, Circle and Ellipse drawing algorithms.
CO2: Implement different 2D Scan conversion & polygon filling algorithms.
CO3: Comprehend 2D Geometric transformations and implement transformation matrices for scaling, shear, rotation, reflection
CO4: Implement different 2D Clipping algorithms
CO5: Comprehend 3D representation of objects, 3D transformation and projection.
CO6: Comprehend and implement the algorithms for B-Spline and Bezier curves

Course code	CSE/PC/B/S/312
Category	Professional Core
Course title	Systems Programming Lab
Scheme and Credits	L–T–P: 0-0-3; Credits: 1.5; Semester – I
Pre-requisites (if any)	

Syllabus:

The system programming laboratory aims to motivate students in developing a fundamental level of understanding on syntax of assembly language programming using SIC, SIC/XE and 8086 instruction sets.

The syllabus of this laboratory also includes developing assembly language programs and hands-on experiences in designing assemblers, macro processor, text editor, linker, loader and device drivers etc.

Course Outcomes:
At the end of this course, each student should be able to:
CO1: Understand syntax of assembly language programming using SIC, SIC/XE and 8086 instruction sets.
CO2: Apply the theoretical fundamentals for solving programming problems
CO3: Develop various solutions on functionality of designing assemblers, macro processor, and text editor.
CO4: Demonstrate how linker and loader create an executable program from an object module created by assembler.
CO5: Develop a basic device driver for hardware interfacing with IO device.

Course code	CSE/PC/B/S/313
Category	Professional Core
Course title	Operating System Lab
Scheme and Credits	L–T–P: 0-1-2; Credits: 2.0; Semester – I
Pre-requisites (if any)	

Syllabus:

The operating system laboratory aims at getting the students prepared for developing modules of an operating system. Along with the theoretical course of Operating System that provides concept of generalized operating system, this lab supplements it by providing students with assignments on different OS management techniques.

The topics in this laboratory covers shell programming in Linux, techniques related to process management including process scheduling, process synchronization, inter process communication and deadlock management. The syllabus of this laboratory also includes developing programs for well known memory management techniques. Security and protection related programming also finds place in the assignment.

Course Outcomes (COs):

At the end of this course, each student should be able to:
CO1: Discuss concept of shell and write and execute shell scripts in Linux.
CO2: Demonstrate Inter process Communication techniques.
CO3: Recognize and Implement process management and synchronization techniques.
CO4: Develop CPU scheduling algorithms and Page replacement schemes.
CO5: Develop technique for secured and controlled access.

Course code	CSE/PC/B/S/314
Category	Professional Core
Course title	Computer Networks Lab
Scheme and Credits	L–T–P: 0-1-2; Credits: 2.0; Semester – I
Pre-requisites (if any)	

Syllabus:

Simulation experiments for MAC, Network, Transport and Application layer protocols – error detection, flow control, multiple access techniques, routing, Socket-TCP, UDP, File Transfer, DNS etc., Network management experiments, configuring, testing and measuring network devices and parameters.

References:

1. Behrouz A. Forouzan, Data Communication and Networking, McGraw-Hill.
2. William Stallings, Data and Computer Communication, Prentice Hall of India.

3. Andrew S. Tanenbaum, Computer Networks, Prentice Hall.
4. Douglas Comer, Internetworking with TCP/IP, Volume 1, Prentice Hall of India.
5. W. Richard Stevens, TCP/IP Illustrated, Volume 1, Addison-Wesley.

Course outcomes:

CO1 : Design and implement error detection techniques within a simulated network environment.
CO2 : Design and implement flow control mechanisms of Logical Link Control of Data Link Layer within a simulated network environment.
CO3 : Design and implement medium access control mechanisms within a simulated network environment using IEEE 802 standards.
CO4 : Design and implement routing protocols within a simulated network environment
CO5 : Analyze protocols and network traffic within a simulated network environment or using network tools.
CO6 : Design and implement various applications using Transport layer protocols and Application layer protocols for its implementation in client/server environment and analyze the performance.

B.C.S.E THIRD YEAR SECOND SEMESTER

Course code	CSE/PC/B/T/321
Category	Professional Core
Course title	Database Management Systems
Scheme and Credits	L–T–P: 3-0-0; Credits: 3.0; Semester – II
Pre-requisites (if any)	

Syllabus:

Introduction: Advantages of DBMS, Various levels of Data Definition and abstraction, Data Independence [2L]
Concepts of Different Database Models, Functional Components of DBMS and Overall Structure of DBMS [2L]
Relational Model: Relation, Attribute, Key, Foreign Key and other Relational Constraints [2L]
Database Design: ER Diagram, Mapping and Participation Constraints, Weak Entity Set, Aggregation, Extended ER diagram, Design of Database Tables from ER/EER Diagram [4L]
Languages: Relation Algebra, Relational Calculus [3L]
Structured Query Language [3L]
Functional Dependency: Concepts of Functional Dependency, Normalization, Multivalued Dependency [5L]
Database Storage: Fixed/Variable Length Record, Ordered/Unordered file and Operations on them [1L]
Indexing: Primary/Clustering/Secondary/Multilevel Index, B/B+ Tree based Indexing, Hashing [3L]
Query Optimization: Search Strategies, Expression level Optimization, Join strategies [2L]

Database Security [1L]
Case Study: Introduction to Oracle Architecture, PL/SQL, Trigger [3L]
Transaction and Recovery: Concept of Transaction and its States, Log based Recovery, Checkpoint [3L]
Concurrency Control: Lock based Protocol, Time Stamp based Protocol, Recoverable Schedule etc. [3L]
Advanced Concepts: Object-oriented database concepts and other query languages [3L]

Books:

1. Fundamentals of Database Systems by E. Navathe, Pearson
2. Database System Concepts by Korth and Silberschatz, McGrawHill
3. Commercial Application Development Using Oracle Developer - 2000 by I. Bayross, BPB

Course Outcomes (COs):

At the end of the course a student will be able to:
CO1: Understand the fundamental concepts of DBMS and relational
CO2: Represent the database using Entity-Relation Model and design the database
CO3: Understand functional dependency and normalize the database
CO4: Interact with database using SQL, PL/SQL, Trigger
CO5: Conceptualize the principles of query optimization, transaction processing, concurrency control, recovery

Course code	CSE/PC/B/T/322
Category	Professional Core
Course title	Internet Technology
Scheme and Credits	L–T–P: 3-0-0; Credits: 3.0; Semester – II
Pre-requisites (if any)	

Syllabus:

Introduction: History of Internet and WWW, Internet Infrastructure, Internet Standards and Authorities. [2L]
Internet Communication Protocols –Transmission Control Protocol/Internet Protocol (TCP/IP) protocol stack suite, IP and protocols for address resolution, Internet control, routing, broadcasting and multicasting, IP Subnetting and Addressing. Advanced protocols, such as DHCP and IPv6. [6L]

End-to-end communication issues associated with TCP, including error control, flow control, congestion control, quality of services (QoS). [8L]

Network management and domain name systems. [4L]

Emerging high level Internet protocols. [4L]

Multimedia Protocols [2L]

World Wide Web: Web applications, Web 2.0, Web 3.0, HTTP [4L]

Client Server Concepts: Client-Server Architecture and Programming In Java, Web Server. [2L]

Web Services: Design and Implementation, Web programming, Javascript, Cookies, Web application frameworks [14L]

Internet Security: Web Security, Web Application Vulnerabilities (https) [6L]

Suggested Readings:

1. TCP/IP Protocol Suite, 4th or higher edition edition by Behrouz A. Forouzan, Mc Graw Hill
2. Internet and World Wide Web How to Program by Harvey M. Deitel, Paul J. Deitel, and T. R. Nieto, 4th Edition, Prentice Hall PTR Upper Saddle River, NJ, USA, 2008
3. Node.js Design Patterns by Mario Casciaro, 2014
4. Expert Spring MVC and Web Flow by Seth Ladd, Darren Davison, Steven Devijver, and Colin Yates, 2006 Apress
5. Cryptography and Network Security, by William Stallings, Fifth Edition, , Prentice Hall, 2010

Course Outcomes:

The students will be able to

CO1: Understand how Internet is controlled and managed through different policy formulation and implementation agencies.
CO2: Describe the protocols and standards adopted by the Internet at the transport layer and networking layer and understand and analyse various performance issues.
CO3: Describe the commonly used application layer services, such as http and https, World Wide Web, and email with an understanding of client-server architecture.
CO4: Analyse and develop Internet applications based on client-server programming model.
CO5: Understand the security threats over the Internet and design and develop technologies to counter the threats.

Course code	CSE/PC/B/T/323
Category	Professional Core
Course title	Compiler Design
Scheme and Credits	L–T–P: 3-0-0; Credits: 3.0; Semester – II
Pre-requisites (if any)	

Syllabus:

Introduction: Formal Languages Theory: Grammars and Language Classification; Analysis-Synthesis model of translation; Structure of a compiler; Issues in compiler design. [2L]

Lexical Analysis: Role of a lexical analyzer; Input buffering, Specification of tokens, Recognition of tokens; Regular expressions; Finite automata, Nondeterministic and deterministic finite automata,

Transitions tables, Acceptance of input strings by automata, Conversion of a Regular Expression to NFA, Conversion of a Regular Expression to DFA; Incorporating a symbol table; Lexical Analyzer Generator. [10L]

Syntax Analysis: Role of a parser, Representative grammars, Context-free grammars, Parse trees, derivations and sentential forms, Ambiguity; [4L]

Top down parsing, Predictive and Recursive descent parsing, Elimination of left recursions, Left factoring, FIRST and FOLLOW sets and their computations, LL(1) grammars, Error recovery techniques; [10L]

Bottom up parsing, Reductions, Handle pruning, Shift reduce parsing; LR parsing, Implementing the parser as a state machine, viable prefixes, Items and the LR(0) automaton; Constructing SLR parsing tables: LR(0) grammars, SLR(1) grammars; Canonical LR(1) items and constructing canonical LR(1) parsing tables; Constructing LALR parsing tables. [10L]

Using Yacc and Lex. [2L]

Semantics and Semantic Analysis: Syntax-directed translation, Attribute grammars, Inherited and synthesized attributes, Dependency graphs, Evaluation orders of attributes, S-Attributed definitions, L-attributed definitions, Syntax-directed translation schemes, Bottom-up evaluation of S-attributed definitions and L-attributed definitions. [6L]

Symbol tables and their relationship to semantic objects; Symbol table implementation: binary trees vs. hashing. [2L]

Intermediate Code Generation: Intermediate languages – Declarations – Assignment Statements – Boolean Expressions – Case Statements – Back patching – Procedure calls. [4L]

Code Optimization: Overview of optimization; Data Flow Analysis; Peephole Optimizations; Constant Folding, Common Subexpression Elimination, Copy Propagation, Strength Reduction. Global Optimization: Loop optimizations; Induction Variable elimination, Optimizing procedure calls – inline and closed procedures. Machine-Dependent Optimization: Pipelining and Scheduling. [2L]

Code Generation: Issues in the design of code generator – The target machine, Run-time storage management, Basic blocks and flow graphs, Liveness and Next-use information, Register allocation and assignment, The dag representation of basic blocks. [4L]

Runtime Environment: Static versus dynamic storage allocation, Names, scopes and bindings; Object lifetimes; Stack allocation; Access to non-local data on the stack; Heap management; Garbage collection. [4L]

Suggested Readings:

1. "Compilers – Principles, Techniques, and Tools"; Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman; Pearson Education.
2. "Compiler Design in C", Allen I. Holub, Prentice Hall.
3. "Crafting a compiler with C", C. N. Fischer and R. J. LeBlanc, Pearson Education.
4. "Compiler Construction: Principles and Practice", Kenneth C. Louden, , Thomson Learning.

Course Outcomes (COs):

The students will be able to:

CO1: Use regular grammar and finite state machines for recognising tokens of different languages.
CO2: Develop and use context free grammars for parsing high level computer languages and develop parsers for analysing source code and capture syntax errors.
CO3: Apply syntax-directed translation for generation of machine-independent intermediate code.
CO4: Store and manage symbol table.
CO5: Apply code optimization techniques to improve run-time performance of executable code with a clear understanding of run-time environment.
CO6: Use modern tools for analysis of regular and context-free languages.

Course code	CSE/PC/B/T/324
Category	Professional Core
Course title	Software Engineering
Scheme and Credits	L–T–P: 3-0-0; Credits: 3.0; Semester – II
Pre-requisites (if any)	

Syllabus:

1.Introduction and Brief Overview – Basic methods and principles used by engineering, including fundamentals of technical communication, measurement, analysis and design.
Some aspects of the engineering profession, including standards, safety and intellectual property are also covered. [2L]

2. Software Model Driven Development Process - Analysis, Design, Testing(traditional practice diagrams such as DFDs and ERDs etc and Object-Oriented Software Engineering – Concept)- Case study with complete examples. [5L]

3. Requirements Engineering – Definition, Analysis, Development, Management; Standards/Guidelines (IEEE-Analysis, Specification, management) and CASE Tools - Case study with complete examples. [3L]

4. Effort and Cost Estimation Techniques - using COCOMO, COCOMO-II (using Lines of code, Object points, Function points) - Case study with complete examples. [4L]

5. Software Architecture - Architectural styles, architectural patterns, analysis of architectures, formal descriptions of software architectures, architectural description languages and tools, scalability and interoperability issues, Web Engineering Architectures - case studies with example. [3L]

6. Software Quality metrics – Product Revision (Maintainability, Flexibility, Test ability); Product Transition(Reusability, Interoperability, Portability); Product Operations(Reliability, Usability, Correctness, Integrity, Efficiency)- Measure Reliability; Availability; Reusability; Measure Software Complexity; - Case study with complete examples. [10L]

7. Software Evaluation Metrics (Supervised and Unsupervised techniques) – Methods of evaluating clustering- Assessing Clustering tendency, Determination number of clusters, Measuring Clustering quality; Methods for estimating a classifier's accuracy: Holdout method, Random subsampling, Cross-validation, Bootstrap. Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity / Precision and Recall, and F-measures - Case study with complete examples. [10L]

8. Software Testing Technique and Strategies – Black Box, White Box, Integrity testing (top down, bottom up, mixed); test case design; System testing (Recovery, Security, Stress, Performance, Regression, Smoke, Verification, Validation, Acceptance etc..)- case study with example. [3L]

Books:

1. Fundamentals of Software Engineering – C. Ghezzi, M. Jazayeri, D. Mandrioli
2. Software Engineering – Sommerville, Pearson
3. Software Engineering – Martin L. Shooman, TMH
4. Software Engineering , A practitioner's approach – Roger Pressman
5. Software Engineering – Rajib Mall

Course Outcomes (COs):

After completion of this course, each student should be able to:

CO1: Understand the concept and the fundamentals of basic methods and principles. Fundamentals of technical communication, measurement, analysis and design.
CO2: Develop model driven process - analysis, design, testing. The requirements definition and analysis phase is critical to keep the development and maintenance costs to a minimum.
CO3: Understand the concept about effort and cost estimation techniques of a software product using tools.
CO4: Explain software architecture. Analysis and formal descriptions of software architectures.
CO5:Understand the concept and able to measure the software quality and evaluation of software product. Performance measure of software product.
CO6: Understand the software testing technique and strategies.

Course code	CSE/PC/B/T/325
Category	Professional Core
Course title	Design and Analysis of Algorithms
Scheme and Credits	L–T–P: 3-0-0; Credits: 3.0; Semester – II
Pre-requisites (if any)	

Syllabus:

- Introduction: Problem solving -- adding 2 n-bit numbers, multiplication as repeated addition. Running time analysis -- recall of asymptotic notation, big-oh, theta, big-omega, and introduce little-oh and little-omega. Worst case and average case complexity **[4L]**

- Divide and Conquer Algorithm design paradigms with illustrative examples: Integer multiplication revisited with an efficient algorithm that motivates and leads into recurrences. Solving recurrences using recurrence trees, repeated substitution, statement of master theorem. Brief recall of Quicksort and merge sort algorithms and its recurrence. [**8L]**

- Basic Graph Algorithms and Analysis: representation of graphs, BFS, DFS, connected components, topological sorting of DAGs, strongly connected components in directed graphs. [**6L]**

- Greedy algorithms- Greedy choice, optimal substructure property, minimum spanning trees -- Prims and Kruskals, Dijkstra' s shortest path , fractional knapsack, and Huffman coding **[8L]**

- Dynamic Programming: 0-1 knapsack (contrasted with the fractional variant), longest increasing subsequence, minimum edit distance and String alignment, matrix chain multiplication [**6L]**

- NP-completeness: reduction amongst problems, classes NP, P, NP-complete, and polynomial time reductions [**4L]**

- Introduction to randomized algorithms [**4L]**

Suggested Readings:

Text Books*:*

1. Introduction to Algorithms, by Cormen, Leiserson, Rivest, and Stein, MIT Press, Third Edition, 2009.

Reference Books**:**

1. Algorithms by Dasgupta, Papadimitrou and Vazirani, McGraw-Hill Education, 2006.
2. Computer Algorithms, by Horowitz, Sahni, and Rajasekaran, Silicon Press, 2007.
3. Algorithm Design, by Kleinberg and Tardos, Pearson, 2005.
4. Algorithm Design, by Goodrich and Tamassia, Wiley, 2001.

Course Outcomes (Cos):

The students will be able to

CO1: Analyze and compare the efficiency of algorithms in terms of running time using asymptotic analysis, and understand how to prove correctness of algorithms

CO2: Be familiar with divide-and-conquer algorithm design techniques and analysis of divide and Conquer algorithms applied to solving various problems like searching, sorting, integer multiplication

CO3: Learn basic graph algorithms and their analysis

CO4: Be familiar with greedy algorithm design technique and its applications in designing various algorithms such as finding shortest paths in a graph, finding minimum spanning trees, Huffman coding etc.

CO5: Learn dynamic programming algorithm design techniques, and its applications in solving some selected real world problems

CO6: Learn basic concepts of randomized algorithms and understand computational tractability of a problem

Course code	CSE/PC/B/T/326
Category	Professional Core
Course title	Artificial Intelligence
Scheme and Credits	L–T–P: 3-0-0; Credits: 3.0; Semester – II
Pre-requisites (if any)	

Syllabus:

Overview: Foundations, Scope, Problems, and Approaches of AI [2L]

Intelligent Agents: Structure of Intelligent Agents, Environments [2L]

Solving Problems by Searching: Graph Search, Uninformed Search, Informed/Heuristic Search, Constraint Satisfaction Search, Stochastic Search (like Hill climbing, Simulated Annealing), Population Based/Evolutionary Search (like Genetic Algorithm, Swarm Optimization), Adversarial Search - Game Playing, Planning as Search, Sample Applications [12L]

Knowledge and Reasoning: Ontology, Foundations of Knowledge Representation and Reasoning, Propositional Logic, First Order Logic, Inference Rules, Unification, Soundness, Completeness, Control Strategies, Resolution-Refutation Systems, Sample Applications [6L]

Reasoning under Uncertainty: Non Monotonic Reasoning Systems, Assumption based Truth Maintenance System, Modal and Temporal Logic, Probabilistic Reasoning (Bayes' Rule, Bayesian Network), Fuzzy Reasoning [4L]
Learning based AI: Learning (Supervised, Unsupervised, Reinforcement), Decision Trees and Random Forests, Neural Learning [6L]

Some Applications of AI [4L]

Ethical and Legal Considerations in AI [2L]

Future of AI [2L]

Books:

1. Artificial Intelligence: A New Synthesis, N. J. Nilsson, Harcourt Asia Pte Ltd, 2000.

2. Artificial Intelligence - A Modern Approach: S. J. Russel and P. Norvig, Pearson Education, 4th Edition, 2020.

3. Principles of Artificial Intelligence: N. J. Nilsson, Narosa, 2002.

4. Artificial Intelligence - Structures and Strategies for Complex Problem Solving: G. F. Luger, NY: Addison-Wesley, 6th Edition, 2008.

5. Essentials of Artificial Intelligence: M. Ginsberg, CA: Morgan Kaufman Publishers, 1993.

6. Artificial Intelligence: E. Rich, K. Knight, and S. B. Nair, McGraw Hill Education, 3rd Edition, 2017.

7. Introduction to Artificial Intelligence and Expert Systems: D. W. Patterson, Pearson Education India, 1st Edition, 2015.

8. Artificial Intelligence - Foundations of Computational Agents: D. L. Poole and A. K. Mackworth, NY: Cambridge University Press, 2nd Edition, 2017.

9. Artificial Intelligence: P. H. Winston, Pearson, 3rd Edition, 1992.

10. Pattern Recognition and Machine Learning: C. M. Bishop, NY: Springer-Verlag, 2006.

Course Outcomes:

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

CO1: Comprehend the concept and objectives of artificial intelligence

CO2: Understand the concept of searching for solving problems, analyse various search algorithms and evaluate and demonstrate their effectiveness

CO3: Comprehend the concept of knowledge representation and reasoning both under ideal situations and under uncertainty and imprecision with applications to theorem proving, answer extraction, real world control applications etc.

CO4: Design and develop various learning based artificial intelligence algorithms and evaluate & demonstrate their applications in solving real life complex problems

CO5: Comprehend the ethical and legal aspects of artificial intelligence

Course code	CSE/PC/B/S/321
Category	Professional Core
Course title	Database Management System Lab
Scheme and Credits	L–T–P: 0-1-2; Credits: 2.0; Semester – II
Pre-requisites (if any)	

Syllabus:

1. Familiarization with basic activities: database creation, constraint specification and interaction with database
2. Programming through PL/SQL

3. Implementation of triggers
4. Developing GUI

Database Management Systems Lab
CO1: Design the database and interact with database using structured query language
CO2: Interact with database through Procedural programming
CO3: Implement triggers
CO4: Develop GUI based database oriented application

Course code	CSE/PC/B/S/322
Category	Professional Core
Course title	Compiler Design Lab
Scheme and Credits	L–T–P: 0-1-2; Credits: 2.0; Semester – II
Pre-requisites (if any)	

Syllabus:

Learning to construct regular expressions for recognising various patterns in different languages.

Learning to use tools for lexical analysis, such as 'lex'.

Designing own lexical analyser for a simple computer language.

Learning to store and manage symbol table.

Designing a parser for parsing context-free languages.

Learning to use a parser tool, such as 'yacc'.

Being acquainted with compiler phases like code optimisation, code generation.

Implementing a mini-project on certain phases of compilers.

Course Outcomes:

The students will:

CO1: Be exposed to compiler writing tools.
CO2: Learn to implement the different phases of compiler.
CO3: Learn to use data structures to efficiently manage symbol table.

Course code	CSE/PC/B/S/323
Category	Professional Core
Course title	Internet Technology Lab
Scheme and Credits	L–T–P: 0-0-3; Credits: 1.5; Semester – II
Pre-requisites (if any)	

Syllabus:

1. Practical experiments on client server applications using TCP/UDP and Web socket protocol.
2. Experiments using web servers to write HTTP request-response services using Java based technologies and node.js.
3. Concept of session management of HTTP applications.
4. Introducing design patterns for implementing web applications.
5. Experiments applying Spring framework along with separate frameworks for developing the front-end of applications.

Course Outcomes:

On completion of this course, students should be able to:

CO1 :Design and implement simple client server applications using TCP/UDP and Websocket protocol.
CO2 :Design and implement HTTP request-response services through web servers
CO3 :Apply the concept of session management to HTTP applications
CO4 :Design web applications following design patterns
CO5 :Compare and implement real-life web applications following different web frameworks

Course code	CSE/PC/B/S/324
Category	Professional Core
Course title	Artificial Intelligence Lab
Scheme and Credits	L–T–P: 0-0-3; Credits: 1.5; Semester – II
Pre-requisites (if any)	

Syllabus:

1. Implement various Uninformed Search Processes (e.g., Breadth First Search, Depth First Search, Depth Limited Search, Iterative Deepening Search, Iterative Broadening Search) considering

a) Tree, b) Graph, c) Water Jug Problem, d) 8-Puzzle Problem, e) Maze Problem

Report *Order of nodes visited* and *Solution Path* for each of the search techniques.

2. Implement Uniform Cost Search using a Graph. Report *Order of nodes visited*, *Solution Path*, and *Solution Cost*.

3. Implement various Uninformed Search Processes (e.g., Best First Greedy Search, A* Search) considering

a) Tree, b) Graph, c) 8-Puzzle Problem and d) Maze Problem

Report *Order of nodes visited* and *Solution Path* for each of the search techniques.

4. Implement single solution based search algorithm (e.g., Hill Climbing Algorithm) for Function Optimization.

5. Explore population based intelligent search method (e.g., Genetic Algorithm) to solve optimization problems.

6. Write a program to solve 4-Queens problem using Adversarial Search.

7. Explore Artificial Neural Network as a Classifier.

8. Write a program to implement Bayesian network for solving prediction problem in Uncertain environment.

9. Use of Fuzziness to handle Uncertainty.

10. Practice Session on Prolog: Use of Facts, Predicates etc. Write various programs on it.

Course Outcomes:

At the end of this course, each student should be able to:
CO1: Apply the fundamental concept of various uninformed and informed search algorithms, design and analyse these algorithms and evaluate & demonstrate their effectiveness on various problems

CO2: Design and develop various advanced intelligent search algorithms and evaluate and demonstrate their applications in solving real life complex problems.

CO3: Implement the concept of knowledge representation and reasoning both under ideal situations and under uncertainty and imprecision

CO4: Design and develop learning based artificial intelligence algorithms (e.g., artificial neural networks) and evaluate and demonstrate their applications