Curriculum Structures of the B. Tech programme in Computer Science and Engineering (CSE) (As per AICTE 2018 guidelines)

Basic Science Courses:

Course Code	Course Name	Credit				CH
		L	T	P	Total	
MS104	Mathematics-I	3	1	0	4	4
MS105	Mathematics-II	3	1	0	4	4
MS205	Mathematics-III	3	0	0	3	3
CO105	Discrete Maths	3	1	0	4	4
PH103	Physics-I	2	0	1	3	4
PH104	Physics-II	2	0	0	2	2
CH103	Chemistry	3	0	1	4	5
BT201	Biology	3	0	0	3	3
	Total-	22	3	2	27	29

HSS and Management Courses:

Course Code	Course Name	Credit				CH
		L	T	P	Total	
EF103	English	2	0	1	3	4
BA201	Economics	3	0	0	3	3
IC361	Accounting and Financial Management	3	0	0	3	3
	HMS Elective-I	3	0	0	3	3
	Total-	11	0	2	12	13

HSS and Management Elective Courses:

Course Code	Course Name	Credit				CH
		L	T	P	Total	
EG4XX	Introduction to Academic Reporting in English	2	0	1	3	4
BA4XX	Fundamentals of Management	3	0	0	3	3
BA4XX	Social Responsibility and Professional Ethics in Engineering	3	0	0	3	3

Engineering Science Courses:

Course Code	Course Name	Credit				CH
CE1XX	Engineering Graphics and Design	1	0	2	3	5
EE1XX	Basic Electrical Engineering	3	0		3	3
EE1XX	Basic Electrical Engineering Lab	0	0	1	1	2
EL1XX	Basic Electronics	2	1	1	4	5
ME1XX	Workshop Practice	0	0	2	2	4
CO103	Introductory Computing	2	1	0	3	3
CO104	Computing Laboratory	0	0	2	2	4
EL2XX	Signals and Systems	2	1	0	3	3
CO218	Data Communication	3	0	0	3	3
CO209	Computing Workshop (SciLab/MatLab/R & Simulator Lab)	0	0	2	2	4
	Total-				26	36

Professional Core Courses:

Course Code	Course Name	Credit				CH
		L	T	P	Total	
CO202	Digital Logic Design	3	0	1	4	5
CO210	Data Structures	3	1	0	4	4
CO216	Formal Languages and Automata	3	0	0	3	3
CO214	Computer Architecture and Organization	3	1	0	4	4
CO215	Computer Architecture and Organization lab	0	0	1	1	2
CO206	Design and Analysis of Algorithm	3	0	1	4	5
CO211	Data Structures using Object Oriented Programming Lab	0	1	2	3	5
CO309	Operating Systems	3	0	0	3	3
CO312	Database Systems	3	0	0	3	3
CO315	Computer Networks	3	0	0	3	3
CO310	Operating Systems Lab	0	0	1	1	2

CO316	Computer Network Lab	0	0	1	1	2
CO313	Database Systems Lab	0	1	1	2	3
CO314	System software and Compiler Design	3	0	1	4	5
CO311	Software Engineering	3	0	0	3	3
CO401	Artificial Intelligence	3	0	0	3	3
CO217	Graph Theory	3	0	0	3	3
CO303	Computer Graphics	3	0	1	4	5
	Total-			10	53	63

Category-wise Break-up:

Course Category	**Credits**
Humanities, Social Science and Management Courses (HSMC)	12
Basic Science Courses (BSC)	27
Engineering Science Courses (ESC)	26
Professional Core Courses (PCC)	53
Professional Elective Courses (PEC)	17
Open Elective Courses (OEC)	12
Projects	14
Total-	**161**

Courses offered in B. Tech. in Computer Science and Engineering

Semester-I

S. No.	Course Code	Course Name	L	T	P	CR	CH
1	CH103	Chemistry	3	0	1	4	5
2	PH103	Physics-I	2	0	1	3	4
3	MS104	Mathematics-I	3	1	0	4	4
4	EE103	Basic Electrical Engineering	3	0	0	3	3
5	EE104	Basic Electrical Engineering Lab	0	0	1	1	2
6	EF103	Communicative English	2	0	1	3	4
7	SE100	Induction Program	-	-	-	-	8
		Total	13	1	4	18	22

Semester-II

S. No.	Course Code	Course Name	L	T	P	CR	CH
1	PH104	Physics-II	2	0	0	2	2
2	MS105	Mathematics-II	3	1	0	4	4
3	CO105	Discrete Mathematics	3	1	0	4	4
4	EC102	Basic Electronics	2	1	1	4	5
5	ME102	Workshop Practice	0	0	2	2	4
6	CO103	Introductory Computing	2	1	0	3	3
7	CO104	Computing Lab	0	0	2	2	4
8	CE103	Engineering Graphics	1	0	2	3	5
		Total	13	4	7	24	31

Semester-III

S. No.	Course Code	Course Name	L	T	P	CR	CH
1	MS205	Mathematics – III	3	0	0	3	3
2	CO202	Digital Logic Design	3	0	1	4	5
3	CO209	Computing Workshop	0	0	2	2	4
4	BA201	Economics	3	0	0	3	3

5	CO210	Data Structures	3	1	0	4	4
6	CO211	Data structures using Object Oriented Programming Lab	0	1	2	3	5
7	EC205	Signals and Systems	2	1	0	3	3
8	ES201	Environmental Science	2	0	1	0	3
		Total	14	3	5	22	30

Semester-IV

S. No.	Course Code	Course Name	L	T	P	CR	CH
1	BT201	Biology	3	0	0	3	3
2	CO218	Data Communication	3	0	0	3	3
3	CO214	Computer Architecture and Organization	3	1	0	4	4
4	CO215	Computer Organization Lab	0	0	1	1	2
5	CO216	Formal Language and Automata	3	0	0	3	3
6	CO206	Design and Analysis of Algorithms	3	0	1	4	5
7	CO217	Graph Theory	3	0	0	3	3
		Total	18	1	2	21	23

Semester-V

S. No.	Course Code	Course Name	L	T	P	CR	CH
1	CO309	Operating Systems	3	0	0	3	3
2	CO311	Software Engineering	3	0	0	3	3
3	CO310	Operating Systems Lab	0	0	1	1	2
4	CO312	Database Systems	3	0	0	3	3
5	CO313	Data base Systems Lab	0	1	1	2	3
6	CO303	Computer Graphics	3	0	1	4	5
7		Elective-I (PEC01)	3	0	0	3	3
8		Open Elective-I (OEC01)	3	0	0	3	3
9	LW301	Indian Constitution (MC- Non Credit)	1	0	0	0	1
		Total	18	1	3	22	25

Semester-VI

S.No	Course Code	Course Name	L	T	P	CR	CH
1	CO314	System Software and Compiler Design	3	0	1	4	5
2		Elective-II (PEC02)	3	0	1	4	5
3		Open Elective-II (OEC02)	3	0	0	3	3
5	CO315	Computer Networks	3	0	0	3	3
6	CO316	Computer Networks Lab	0	0	1	1	2
7	IC361	Accounting and Financial Management	3	0	0	3	3
8	CO317	Project-I (using SE perspective)	0	0	2	2	4
		Total	15	0	5	20	25

Semester-VII

S. No.	Course Code	Course Name	L	T	P	CR	CH
1	XXxxx	* HSS/Management Elective	3	0	0	3	3
2	CO401	Artificial Intelligence	3	0	0	3	3
3		Elective-III (PEC03)	3	0	1	4	5
4		Elective-IV (PEC04)	3	0	0	3	3
5		Open Elective-III (OEC03)	3	0	0	3	3
6	CO402	Project-II	0	0	4	4	8
7	CT430	Essence of Indian Traditional Knowledge (MC- Non Credit)	1	0	0	0	1
		Total-	15	0	6	20	26

Semester-VIII

S.No.	Course Code	Course Name	L	T	P	CR	CH
1		Elective-V (PEC05)	3	0	0	3	3
2		Open Elective-IV (OEC04)	3	0	0	3	3
3	CO403	Project-III	0	0	8	8	16
		Total	6	0	10	14	22

Mandatory Non-credit Courses:

S. No.	Course Code	Course Name	Schedule
1	SE100	Induction Programme	Semester I
2	CO404	Summer Internship /Industrial Training	After Semester VI
3	CO405	Comprehensive Written Exam	Semester VII/Semester VIII
4	ES201	Environmental Science	Semester III
5	LW301	Indian Constitution	Semester V
6	CT465	Indian Traditional Knowledge	Semester VI

Semester-wise Break-up:

Semester	Category-wise Credit Distribution							Total Credits
	HSMC	BSC	ESC	PCC	PEC	OEC	PCE	
I	3	11	4	0	0	0	0	18
II	0	10	14	0	0	0	0	24
III	3	3	5	11	0	0	0	22
IV	0	3	3	15	0	0	0	21
V	0	0	0	16	3	3	0	22
VI	3	0	0	8	4	3	2	20
VII	3	0	0	3	7	3	4	20
VIII	0	0	0	0	3	3	8	14
Total	**12**	**27**	**26**	**53**	**17**	**12**	**14**	**161**

Professional Elective Courses (PEC):

PEC01, PEC02, PEC03, PEC04, PEC05 will be offered from the following list of Elective courses from the Department. Some of the elective courses are already running in the B.Tech. programme and some of them are added newly.

Course Code	Course Name	L-T-P	Credit
CO304	Principles of Programming Languages	3-0-0	3
CO318	Cryptography	3-0-0	3
CO432	Information Theory and Coding	3-0-0	3
CO319	Statistical Modelling and Applications	3-0-0	3

CO423	Web Technology	3-0-1	4
CO306	Embedded Systems	3-0-1	4
CO426	Advanced Computer Architecture	3-0-1	4
CO422	Theory of Computation	3-1-0	4
CO406	Distributed Systems	3-0-1	4
CO509	Computer Vision & Image Processing	3-0-1	4
CO512	Parallel Programming	3-0-1	4
CO513	Fundamentals of Speech Processing	3-0-1	4
CO514	Machine Learning	3-0-1	4
CO515	Knowledge Representation and Reasoning	4-0-0	4
CO516	Advanced Algorithms	4-0-0	4
CO517	Virtual and Augmented Reality	3-0-1	4
CO518	Cloud Computing	3-0-0	3
CO504	Natural Language Processing	3-0-0	3
CO519	Internet of Things	3-0-0	3
CO520	Software Defined Networking and Network Function Virtualization	3-0-0	3
CO521	Computational Geometry	3-0-0	3
CO522	Bioinformatics	3-0-0	3
CO523	Quantum Computing	3-0-0	3
CO524	Linear Optimization	3-0-0	3
CO505	Advanced Database Management System	3-0-0	3
CO503	Fuzzy Logic and Neural Networks	3-0-0	3
CO435	Mobile Computing	3-0-0	3
CO525	Data Mining	3-0-0	3
CO526	Operation Research	3-0-0	3

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

- Operating systems and OS design issues
- OS based programming fundamentals
- be familiar with OS system software design concepts
- be able to design and develop OS features
- 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 non-contiguous 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:

3. Operating systems: Concepts and Design, M. Milenkovic (McGraw Hill, 2001)
4. Modern Operating Systems, A. S. Tanenbaum (Pearson, 2009)
5. Design of the Unix Operating System, M. J. Bach (Prentice Hall of India, 1986)
6. Operating Systems: Three Easy Pieces, Remzi and Andrea Arpaci-Dusseau 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)

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

- to grasp knowledge for implementation of operating system software features,
- to understand the working of an operating system kernel,
- to master on using system level programming especially the kernel coding using C and java /C++ languages,
- to master on develop algorithm for operating system design
- to grasp knowledge for implementing file system concepts
- 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 & Vol-2, 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

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:

- Familiar with overall concepts involved in the software development project.
- Able to understand fundamental concepts of requirements engineering and Analysis Modelling.
- Able understand the various software design methodologies
- Able to understand various testing and maintenance measures.
- 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

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:

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

Database Systems — C0 312

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:

- The students should clearly understand the concepts of database systems, their advantages, and applications.
- The students should be able to design a database system for a given database problem.
- 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:

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

References:

- Database Management Systems, 3rd Edition, - Raghu Ramakrishnan, Johannes Gehrke, McGraw Hill, 2014

- An Introduction to Database Systems, 8th Edition, C.J Date, Pearson, 2003
- Fundamentals of Database Systems, by Leon & Leon, Tata McGraw Hil, 2008
- SQL & NoSQL Databases, Meier, Andreas and Kaufmann, Michael, eBook, Springer 2019
- Getting Started with NoSQL, Gaurav Vaish, Packt Publishing, March 2013
- Hadoop: The Definitive Guide, 4th Edition, Tom White, O'Reilly Media, Inc., 2015

Database Systems Lab C0313

0 - 1 - 1: 2 Credits: 3 Hours *Prerequisites:* CO312

Course Outcomes:

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

- Create and manage a database using RDBMS like Oracle.
- Perform queries on the database.
- Build user-defined functions and procedures using PL/SQL.
- 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, Color-display 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-

- The task of translation of assembly language and high-level language programs into machine language programs,
- Description of languages using formal grammars and use of tools for recognition of input strings.
- Automate the task of parser construction and produce translations of input.
- Implement symbols tables.

- The run-time memory and execution environment of programs.
- The task of code optimization.
- The scope of a compiler in leveraging the advances of computer architecture.
- The various types of linking of program modules.
- The tools available for program debugging, version control, building large executable programs.
- 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 -

- Understand the need of a layered Architecture for Computer Network.
- Understand the key functions performed by different widely known protocols at Network, Transport and Application Layers.
- Configure a network with Router, ARP, DHCP, DNS, and Gateway etc.
- Apply mathematical foundations to solve computational problems in computer networking
- Understand the basic working behavior of TCP and UDP.
- Know the use of different networking devices.
- Understand the needs of network security and usages of different security mechanisms.
- Understand the needs and protocols used in different network applications
- 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), Basic Programming, OS

Course Outcomes:

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

- Apply packet sniffing tools to capture packets and analyse
- Write client-server applications in C/C++/Java
- Use different OS tools to configure network and network protocols
- Set up LAN using networking devices
- Use network simulators to study behaviour of different protocols
- 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:

Online and weblink reference manuals for network simulators

Principles of Programming Languages

CO304

3-0 -0:3 Credit : 3 Hours

Prerequisites: CO103 (Introductory Computing), CO104 (Computing Laboratory)

Abstract: This course intends to allow the students to understand the basic concepts underlying different programming languages and their design tradeoffs. The course covers the pragmatic aspects of programming languages that requires a basic knowledge of programming language translation and runtime features such as storage allocation. These topics strengthen students' grasp of the power of computation, help students choose the most appropriate programming model and language for a given problem, and improve their design skills.

Course Outcomes:

- Compare programming languages
- Describe and analyze the main principles of imperative, functional, object oriented and logic oriented programming languages
- Recite the high points of programming language history
- Read the central formalisms used in the description of programming languages
- Assess programming languages critically and in a scientific manner

Course Contents:

Overview of programming languages: History of programming languages, Brief survey of programming paradigms such as Procedural languages, Object-oriented languages, Functional languages, Declarative, non-algorithmic languages, and Scripting languages, Effects of scale on programming methodology, Issues as space efficiency, time efficiency, safety, and power of expression.

Virtual machines: The concept of a virtual machine, hierarchy of virtual machines, Intermediate languages.

Language translation: Comparison of interpreters and compilers, Language translation phases, Machine-dependent and machine-independent aspects of translation.

Declarations and types: The conception of types as a set of values together with a set of operations, Declaration models, Overview of type-checking, Garbage collection

Abstraction mechanisms: Procedures, functions, and iterators as abstraction mechanisms, Parameterization mechanisms (reference vs. value), activation records and storage management, type parameters and parameterized types, modules in programming languages.

Functional programming: Overview and motivation of functional languages, recursion over lists, natural numbers, trees, and other recursively-defined data, pragmatics (debugging by divide and conquer; persistency of data structures), closures and uses of functions as data (infinite sets, streams)

Logic programming: Prolog

Type systems: Data type as set of values with set of operations, Data types such as elementary types, product and co-product types, algebraic types , recursive types, arrow (function) types, and parameterized types, type-checking models, semantic models of user-defined types such as type abbreviations, abstract data types, and type equality, typed-Lamda Calculus, type inference using ML, OCAML or Haskell.

Programming language syntax and semantics: General problem of describing syntax and semantics, formal methods of describing syntax - BNF, EBNF for common programming languages features, parse trees, ambiguous grammars, attribute grammars, overview of formal semantics, informal semantics, denotational semantics, axiomatic semantics, and operational semantics.

Programming language design: General principles of language design, design goals, typing regimes, data structure models, control structure models, and abstraction mechanisms.

Exception Handling: Exception handling in various languages, programming events.

Text Books:

1. Programming Languages-Design and Implementation, TW Pratt, MV Zelkowski, TV Gopal, 4th Edition, Pearson Education India, 2006
2. Programming Languages-Principles and Practice, K. C. Louden, K.A.Lambert, 3rd Edition, Cengage Publications, 2012
3. Programming Languages- Concepts and Constructs, , R. Sethi, 2nd Edition, Pearson Education, 2006

Reference Books:

1. Fundamentals of Programming Languages, Ellios Horowitz, 2nd Edition, Galgotia Publications
2. Concepts of Programming Languages, R.W. Sebesta, 10th Edition, Pearson Education India, 2013

Cryptography — CO318

3-0-0: 3 Credits : 3 Hours

Prerequisites: CO105(DM), CO206 (DAA)

Abstract: In today's world of the internet, the most basic objective is to ensure secure communication across an insecure channel. The art and science of establishing confidentiality, integrity and authenticity of communication is recognized as cryptography. This course is an introduction to modern cryptography. This course introduces basic concepts in cryptography from both theoretical and practical perspectives.

With the emergence of ideas such as power, battles, politics and supremacy, a wide scope of financial, legal, and social cryptographic applications has emerged, from using a credit card on-line or sending an encrypted email, to more ambitious goals of electronic commerce, electronic voting, contract-signing, database privacy, and so on. The most important characteristic of modern cryptography is its *rigorous*, *scientific approach*, based on firm complexity-theoretical foundations. This course will cover how cryptography works, how security is analysed theoretically, such as symmetric-key encryption, stream ciphers, block ciphers, message authentication codes, asymmetric encryption (RSA- and discrete-log-based), and digital signatures.

Course Outcomes:

Towards the end of the course the student will be able to-

- identify, conceptualize, and rigorously formalize the concept of secure communication
- design and analyse security protocols including asymptotic efficiency and provable security
- recognize and explain aspects of number theory which are relevant to cryptography
- identify, explain, and apply cryptographic techniques like key management, digital signatures, digital certificates, and a Public-Key Infrastructure (PKI) to various disciplines in information science.

Course Contents:

- Overview of cryptography. Cryptanalysis, Brief history. Classical and modern cryptography
- Number Theory: gcd, divisibility, euclidean algorithm, extended euclidean algorithm, congruences, modular arithmetic, Chinese Remainder Theorem residue classes, reduced residue systems, Groups, quadratic residues, and finite fields, congruences, modular arithmetic, Computing with large numbers, Algorithms for finding gcd, primality testing and factoring.
- Basic symmetric-key encryption, perfect secrecy, Shannon's definition of

  perfect secrecy, Shannon's Theorem, One-time pad, stream ciphers - LFSR based stream ciphers, RC4, block ciphers – DES, AES, Different modes of operation of Block Cipher – CBC, CFB, Counter Mode, OFB
- Pseudorandom Generators (PRG); Pseudo Random Functions (PRF); Pseudo Random Permutations (PRP); security against Ciphertext Only Attacks (COA); Known

Plaintext Attacks (KPA), chosen plaintext attacks (CPA), chosen ciphertext attacks (CCA)

- Message integrity: definition and applications, MAC, Collision resistant hashing, SHA, HMAC, Authenticated encryption: security against active attacks, session setup using a key distribution center (KDC)
- Public key cryptography, Cryptography using arithmetic modulo primes, Diffie-Hellman key exchange protocol, CDH and discrete-log assumptions, Public key encryption, RSA, ElGamal encryption, limitations of RSA and ElGamal PKCs and various attacks; CCA security
- Digital signatures: Digital signatures: definitions and applications, signature using RSA, Hash based signatures. certificates, certificate transparency, certificate revocation.
- Security Protocols: Identification protocols, Password protocols, salts; one-time passwords (S/Key and SecurID); challenge response authentication, Authenticated key exchange and SSL/TLS session setup, HTTPs, SSH, Zero knowledge protocols,
- Secure Multi-party computation, Cryptography in the age of quantum computers

Text Books:

1. Jonathan Katz and Yehuda Lindell, Introduction to Modern Cryptography, Chapman and Hall/CRC Press, 2nd edition, 2018
2. Rafael Pass and Abhi Shelat, A Course in Cryptography, 3rd edition, 2010

Reference Books:

1. O. Goldreich, Foundations of Cryptography, CRC Press (Low Priced Edition Available), Part 1 and Part 2, 1st edition, 2009
2. D. R. Stinson and M. Paterson, Cryptography: Theory and Practice, CRC Press, 4th Edition, 2018
3. Mike Rosulek. The Joy of Cryptography, 2020
4. Dan Boneh and Victor Shoup. A Graduate Course in Applied Cryptography, 2020
5. Nigel Smart. Cryptography: An Introduction, McGraw-Hill College, 2004

Information Theory and Coding CO432

3-0-0: 3 Credits: 3 Hours

Prerequisites: MS105, MS205

Abstract:

This course is about how to measure, represent, and communicate information efficiently. Information Theory has been the driving force behind the revolution in digital communication and has led to various practical data compression and error correcting codes that meet the fundamental theoretical limits of performance. "Information theory", the breakthrough work of Claude Shannon and Warren Weaver, provided the basis to build the internet, digital computers and telecommunications systems. This course will study how information is measured in terms of probability and entropy, and the relationships among conditional and joint entropies; how these are used to calculate the capacity of a communication channel, with and without noise; coding schemes, including error correcting codes; how discrete channels and measures of information generalise to their continuous forms; the Fourier perspective; and extensions to wavelets, complexity, compression, and efficient coding of audio-visual information

Course Outcomes:

Towards the end of the course the student would understand how to -

- calculate the information content of a random variable from its probability distribution
- define channel capacities and properties using Shannon's Theorems
- construct efficient codes for data on imperfect communication channels
- generalise the discrete concepts to continuous signals on continuous channels
- understand Fourier Transforms
- describe the information resolution and compression properties of wavelets

Course Contents:

- A brief review of probability, uncertainty, information, Concepts of randomness, redundancy, compressibility, noise, bandwidth, and uncertainty, Ensembles, random variables, marginal and conditional probabilities
- Entropies as measures of information. Marginal entropy, joint entropy, conditional entropy, and the Chain Rule for entropy. Mutual information between ensembles of random variables.
- Source coding theorem; prefix, variable-, and fixed-length codes, Noiseless coding, Huffman coding and its optimality, Kraft and McMillan's inequality, Shannon-Fano code, Elias code, Arithmetic coding and universal coding, Error correcting codes
- Algebraic codes-Linear Block codes, Cyclic codes-BCH codes, perfect code, galley codes, Finite geometry codes, Hadamard codes, Maximal distance separable codes, sphere packing and singleton bounds.
- Continuous information; density; noisy channel coding theorem, Fourier theorems; transform pairs. Sampling; aliasing, Gabor-Heisenberg-Weyl uncertainty relation.
- Lossless compression and cryptographic codes, algebraic error correction, Kolmogorov complexity.

Text Books:

1. Raymond W. Yeung, A First Course in Information Theory, Springer, 2002

2. Thomas M. Cover and Joy A. Thomas, Elements of information theory, Wiley, 2nd Ed., 2006.
3. Simeon Ball, A Course in Algebraic Error-Correcting Codes, Springer Nature Switzerland AG, 2020

Reference Books:

1. Claude Shannon, A mathematical theory of communication, Bell systems Tech Journal. Vol. 27, pp. 379–423, 623–656, July, October 1948.
2. David MacKay, Information theory, inference, and learning algorithms, Cambridge University Press 2003.
3. V. Guruswami, A. Rudra, and M. Sudan, Essential Coding Theory, University at Buffalo 2014

Statistical Modelling and Applications — CO319

3-0 -0: 3 Credits : 3 Hours — *Prerequisites:* MS205, CO105

Abstract

In today's world, Data is constantly growing in volume, variety, uncertainty. Uncertainty in data arises due to the presence of noise that adds errors to the original values. Uncertain data is found in abundance today on the web, in sensor networks, finance, urban informatics, and business informatics, meteorology, genomics, complex physics simulations, biology. This course relates to multiple fields, and techniques from various disciplines — probability theory, computer science, optimization, statistics, and many more. Statistical model enables the compact representation and manipulation of exponentially large probability distributions. That allows them to efficiently manage the uncertainty and partial observability that commonly occur in real-world problems.

This course shows the way to construct an effective model, by learning from data that is only an approximation of our past experience. **Representation, inference, and learning** are the pillars of modeling any real-life situation. We discuss random variables, probability distribution for acquiring and **representation** of real-life data. **Inference** is about designing algorithms to be applied on the constructed model or representation to reach a desired conclusion. With newer experiences we **learn** and improve our model so that we don't have to change our **inference** algorithms always. On the whole, we will learn to put probability distributions on real life experiences, learn them from data and do reasoning with them.

Course Outcomes:

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

(a) Understand basic notions of discrete and continuous probability.
(b) Understand the philosophy behind basic methods of statistical inference, and the role that sampling distributions play in those methods.
(c) perform correct and meaningful statistical analyses of simple to moderate complexity; and

(d) Have a first level understanding of Monte Carlo simulation.

Course Contents:

Univariate and Multivariate Distributions

- Probability mass, density, and cumulative distribution functions
- Weak and Strong law of large numbers, Central Limit Theorem
- Parametric families of distributions
- Expected value, variance, conditional expectation
- Applications of the univariate and multivariate Central Limit Theorem
- Probabilistic inequalities
- Markov chains

Sampling, Estimation and Model Building

- Random samples, sampling distributions of estimators
- Statistical inference: Methods of Moments and Maximum Likelihood, confidence interval and testing of hypotheses, tests of significance, chi-square tests, P-values
- Introduction to multivariate statistical models: regression and classification problems, Cluster analysis, principal components analysis
- The problem of overfitting; model assessment

Computer science and engineering applications (Case study on one or more of the following**)**

- Data mining
- Network protocols, analysis of Web traffic
- Computer security
- Software engineering
- Computer architecture, operating systems, distributed systems
- Bioinformatics
- Machine learning

Laboratory experiments:

Programming using the open-source, platform-independent programming language R or the MATLAB package.

Text Books:

- K. Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. Wiley, New York, 2001.
- M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis, Cambridge,* 2005.
- N. Matloff. *A Course in Probabilistic and Statistical Modeling in Computer Science*. http://heather.cs.ucdavis.edu/~matloff/132/PLN
- N. Matloff, *The Art of R Programming: A Tour of Statistical Software Design , 2011.*

- Alan Agresti, *An Introduction to Categorical Data Analysis*, Wiley, New York, 2013

Reference Books:

- *D. Koller & N. Friedman*, Probabilistic Graphical Models: Principles and Techniques, *MIT Press**, 2009***
- M. Baron, *Probability and Statistics for Computer Scientists*, Chapman and Hall/CRC, 2007.

Computational Geometry — CO521

3-0-0: 3 Credits : 3 Hours — *Prerequisites:* MS205, CO105

Abstract

Computational geometry is the study of the representation and storage of geometric data and relationships, and the design, implementation and analysis of computational algorithms that operate on geometric data to answer questions of practical interest. This course introduces students to the essentials of geometric algorithms and presents an in-depth study of the fundamental geometric structures and techniques used in this field.

Course Outcomes:

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

(a) improve problem-solving skills and algorithm design techniques using gometry.
(b) Study the efficiency of geometric algorithms
(c) Design and analyze algorithms for the efficient solution of numerous geometric problems that arise in other application areas such as astronomy, geographic information systems, CAD/CAM, data mining, graph drawing, graphics, medical imaging, metrology, molecular modeling, robotics, signal processing, textile layout, typography, video games, vision, VLSI
(d) use and manipulate several geometric data structures
(e) provide an insight into how one can recognize the inherent geometric properties of a particular application domain.

Course Contents:

- Convex hull
- Line segment intersection
- Triangulation
- Linear programming
- Range search
- Point location
- Voronoi diagram

- Arrangement and duality
- Visibility graph
- Well separated pair decomposition
- VC-dimension, ε-approximation, and ε -nets.
- Surfaces: reconstruction, surface simplification, Mesh generation

Text Books:

- M. de Berg, M. Van Kreveld, M. Overmars, and O. Schwarzkopf, Computational Geometry: Algorithms and Applications (3rd Edition), Springer, 2008.
- F. Preparata and M. Shamos, Computational Geometry, Springer-Verlag, 1985.
- M. Mitzenmacher and E. Upfal. Probability and Computing: Randomized Algorithms and Probabilistic Analysis, Cambridge, 2005.

Reference Books:

- S.L. Devadoss and J. O'Rourke, Discrete and Computational Geometry, Princeton University Press, 2011
- J. O'Rourke, Computational Geometry in C, 2nd ed., Cambridge Univ. Press, 1998.
- Boissonnat, J.-D.; Yvinec, M, Algorithmic geometry - Cambridge University Press , 1997.
- K. Mulmuley, Computational Geometry: An Introduction Through Randomized Algorithms, Prentice Hall, 1994.

Web Technology CO423

3- 0-1: 4 Credits : 5 Hours

Prerequisites: CO103 (Introductory Computing), CO104 (Computing Laboratory)

Abstract: This course provides a basic overview and understanding of many key Web technologies. It starts with understanding the basics of the Internet, thereafter, followed by delving into the underlying technologies such as HTTP, FTP, and middleware such as CORBA, DCOM etc. that supports the smooth functioning of any Internet applications. Alongside, fundamentals on web programming such as HTML, XHTML, Cascading Style Sheets (CSS), Extensible Markup Language (XML), Extensible Stylesheet Language (XSL), XPATH, XSLT are also covered, and describes how scripting, such as JavaScript, jQuery,

and AJAX, works in dynamic websites. The course also discusses browser and server architectures, Web and application servers, Hypertext Preprocessor (PHP), Python, Common Gateway Interface (CGI) and Web security.

Course Outcomes:

Towards the end of the course the student would -

- Understand key Internet technologies supporting the Internet applications.
- Implement interactive web page(s) using HTML, CSS and JavaScript.
- Implement dynamic web page(s) using AJAX, jQuery, JavaScript, PHP etc. and database connectivity.
- Describe and differentiate different Web Extensions and Web Services.
- Implement at least one web security mechanism using PHP, JavaScript, or Python.

Course Contents:

Basics Of Internet: Computer network, Network connectors, Network software, LAN,CAN, MAN, WAN, Introduction to Internet, World Wide Web (WWW), Internet Protocols, Web browser, Web server, Internet services, Web services, DNS, Virtual hosting.

Internet Details: Internet topology, Virtual High Speed Backbone Network Service (vBNS), Network Access Providers (NAS), Internet Service Provider (ISP) and architecture, Internet communication protocols such as HTTP and its different versions, FTP, SMTP, POP, MIME, Email Privacy such as Pretty Good Privacy (PGP), and Privacy Enhanced Email (PEM).

Client/Server Computing: What is C/S Computing, Fat client VS Fat Servers, N-tiered Software Architecture, Middleware, Distributed Object Models such Common Object Requests Broker Architecture (CORBA), Distributed Component Object Model (DCOM), Java Remote Method Invocation (JRMI) and Enterprise Java Bean (EJB).

Markup Languages And Their Grammars: SGML, DTD Resource; HTML, XHTML, CSS, XML, XSL, Query Languages for XML, XML validator script.

Client-Side programming: HTML,CSS, AJAX, JavaScript, jQuery

Server-Side programming: PHP, Python, CGI.

Overview of Java, JAVA Applet, JAVA Servlets, JSP.

Web Service: JAX-RPC-Concepts-Writing a Java Web Service-Writing a Java Web Service Client-Describing Web Services: WSDL- Representing Data Types: XML Schema-Communicating Object Data: SOAP Related Technologies-Software Installation-Storing Java Objects as Files-Databases and Java Servlets.

Web Browser: Browser Architecture, Document Object Model (DOM), DOM Tree, Render Tree, Rendering Engine and its use on various browsers, case study on browser architectures of Mozilla Firefox, Google Chrome, and IE.

Web Server Apache Architecture: Web Server Architecture, Server Features, Configuration of Apache and IIS.

ASP and JSP Search Engines; Web Database Connectivity.

Interface to database: CGI, JDBC

Web Security: Web security threats, Firewalls, Proxy Servers, Cryptography, Digital Signature, Digital Certificates, Secure Socket Layer (SSL), S-HTTP, Secure Electronic Transaction (SET), 3D Secure Protocol.

Practical Topics:

- Implement a basic static web page(s) using HTML and CSS, thereafter, converting it into an interactive web page(s) using JavaScript.
- Implement dynamic web page(s) using AJAX, jQuery, JavaScript, PHP etc. and database connectivity.
- Implement client and server-side scripting for creating dynamic web pages(s).
- Implement at least one web security mechanism using PHP, JavaScript, or Python.

Text Books:

1. Web Technologies: A Computer Science Perspective by Jaffrey C. Jackson, Prentice Hall, 2007
2. Web Technologies: TCP/IP to Internet Application Architecture by Achyut S. Godbole and Atul Kahate, Tata McGraw-Hill Education, 2003, 5th Reprint 2006

Reference Books:

1. Dynamic Web Publishing Unleashed, Shelly Powers et al., 2nd Edition, Sams, 1997
2. Java 1.2 Unleashed, Jamie Jaworski, 4th Edition, Sams, 1998
3. CGI by Example, Jeffry Dwight et.al., Que, 1996.
4. Using Active Server Pages, Scot Johnson et.al., Que, 1997

Embedded Systems CO306

3-0-1: 4 Credits : 5 Hours Prerequisite: CO214(CAO), CO309(OS)

Abstract:

This course is an introduction to embedded system technology and provides hands on experience in embedded system design and implementation using the ARM processor/8051 microcontroller. It provides basic knowledge on embedded system design issues and real time systems. The course encourages students to explore the scope of embedded systems in everyday life and come up with innovative ideas. It includes lectures, laboratory work and a group project.

Course Outcomes:

Towards the end of the course the student would …

-be familiar with the basic technology behind embedded computing systems

-be well conversant with Embedded Systems and ES design issues

-be familiar with Real Time Systems and design issues

-be able to design, simulate and develop intelligent products and systems

Course Contents:

Introduction to embedded systems: Why ES? ES classification and cores, Difference between GPC and ES, ES applications, Components and characteristics of ES, Examples of ES, ES Design metrics, Typical ES architecture, ES Technology, Future & scope of ES.

ES processor design: Categories of ES processor, Processor technologies, Processor architecture, SPP and GPP design, Advantages and disadvantages of using SPPs and GPPs in an ES.

ES Peripherals: Standard SPP or peripherals, Peripheral categorization and classification, User peripherals- GPIO, Timers & Counters, Watchdog Timers, RTC, UART, PWM, LCD controllers, Keypad controllers, ADC, Stepper Motor controllers.

Memory: Classification and technologies-RAM (SRAM, DRAM, PSRAM, NVRAM) & ROM (mask-PROM, OTP-ROM, EPROM, EEPROM, OTP-EPROM, UV-EPROM), Flash, SD-MMC, Advanced RAM, Memory management-cache memory, Cache mapping, Software overlays, Virtual memory.

Interfacing: Basic Terminology, ISA Bus Protocol, Basic protocol concepts, Microprocessor Interfacing: I/O Addressing (port based, bus based), Interrupts, DMA and DMA controller, Programmable priority controller, Bus Arbitration, Advanced communication principles, Communication protocols: Serial, parallel, wireless.

Design and implementation of an ES: Embedded system design, SDLC, Introduction to a simple Digital Camera as an example ES, Basic functions, Designer's perspective, Various Design implementations, Comparison of such implementations.

Sensors and their use in ES: Role of sensors in ES, Sensor interfacing and signal conditioning, Sensor classification, Criteria for choosing a sensor, Types of sensors (temperature sensor, Ultrasonic sensor, LDRs, IR sensor, etc.) and their working principles.

Programmable System Peripherals: AHB and VPB, Programming peripherals in ES, System peripherals: On chip Flash, SRAM memory, External bus interface, PLL, VLSI peripheral bus divider, Power Control, Interrupt systems.

Low Power Computing: What is LPC? Motivation for LPC especially in ES, Power dissipation sources, Static and dynamic power dissipation, Power modelling and estimation, Voltage/frequency scaling, Power reduction in FSM, Power reduction in datapath, Low power memory optimization.

Embedded Software Architectures: Simple and complex architectures, Round Robin, Round Robin with interrupt, Function Queue scheduling architecture, RTOS architecture, Selecting an architecture.

Real Time Systems and RT Operating System: What is a RT system? RT system applications, Basic model, Characteristics, Classification of RT tasks, Tasks and task states, What is RTOS?, RTOS vs OS, RTOS features, RTOS scheduler, Shared data problem, Semaphores and Shared Data, Re-entrant functions, Priority Inversion problem, Scheduling algorithms in RTOS.

8051 Microcontroller/ARM Processor: Introduction to the Architecture, Addressing modes, brief overview of the instruction set, I/O port programming (I/O ports and their functions, bit addressability), Timer programming (timer modes, timer as counter), Serial Port Programming, Interrupts, 8255 PPI, Interfacing to common peripherals.

Practical topics (Demonstration of C/Assembly language programs using microC/Keil software for the 8051µC/ ARM-7 processor**):**

(i) LED on/off control via I/O ports

(ii) Blinking of LEDs in ascending and descending order

(iii) LED control via push buttons

(iv) Display of numbers on 7-segment LED

(v) Displaying data from push buttons on 7-segment LED

(vi) Display of 2-digit number using two 7-segment LEDs

(vii) Interfacing a keypad to AT89S52 microcontroller

(viii) Display message on LCD

(ix) Interfacing ADC to AT89S52 microcontroller

(x) Display of ADC data on PC hyperterminal

(xi) Display temperature from temperature sensor on LCD

(xii) Control of DC motor via microcontroller

Mini Project (Samples):
Display of real time on LCD, Traffic Light System, Automatic turn-on/off of fan and light, automatic control of street lights, object detection, temperature measurement and display, automatic watering of plants, water tank overflow control system, etc.

Text Books:

(i) Embedded System Design: a unified hardware/software introduction, Frank Vahid and Tony Givargis, Wiley, 3rd edition, 2006.

(ii) An Embedded Software Primer, David E. Simon, Addison-Wesley Professional, 13th printing, 2004.

(iii) The 8051 Microcontroller and Embedded Systems, Mazidi, Mazidi, McKinlay, Pearson Education Ltd., 2014.

Reference Books:

(i) Computers and Components, Wayne Wolf, Elsevier, 3rd edition, 2012.

(ii) The 8051 Microcontroller based Embedded Systems, Manish K Patel, Mc Graw Hill, 1st edition, 2014.

(iii) The Insider's Guide to the Philips ARM7-Based Microcontrollers, T. Martin, Hitex (UK) Ltd., 1st edition, 2005.

(iv) Embedded and Real Time Operating Systems, K C Wang, Springer, 1st edition, 2017.

(v) Embedded Software Development with C, Kai Qian, David den Haring and Li Cao, Springer, 1st edition, 2009.

Advanced Computer Architecture CO426

3-0-1: 4 Credits : 5 Hours Prerequisite: CAO

Abstract: CO426 is an advanced level course for undergraduate students in the field of computer architecture at Tezpur University. This course is designed to teach the students about the state-of-the-art in computer architectures and its current trends. Students will gain the knowledge of different architectural developments to improve the efficiency of a computer system and develop the skill to measure the performance of a computer system. They will understand the performance bottlenecks in typical Von-Neumann based architecture and will be familiar with how to exploit different kinds of parallelism possible in the underlying execution model. Students will learn what are the architectural features used in Multiprocessor and Multicore based systems and associated synchronization and Cache Coherence Issues. This will make the students understand how memory hierarchy design can influence the performance of a computer system, what are the different design trade-offs in Cache Memory and Virtual Memory design. Furthermore, they will be familiar with architectures used in Mobile devices, Cloud, and high-performance computing devices. Students will be trained to develop parallel programming skills with OpenMP and CPU-GPU combined environments.

Course Outcomes:

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

- Understand the state-of-the-art in computer architectures and processing
- Evaluate the performance of computer Systems

- Determine the key architectural elements to improve the performance of a computer system
- Understand the Instruction Level Parallelism and the techniques to handle different types of data and control hazards
- Design Memory Hierarchy for improving performance of computer system
- Understand the Vector Processor and GPU Architecture for exploiting parallelism
- Understand the architectures used in mobile, cloud and high-performance computing devices
- Write programs for Parallel Processors and CPU-GPU environment

Course Contents:

- **Introduction**: Defining Computer Architecture, Flynn's Classification of Computers, Metrics for Performance Measurement, Parallel programming, and performance issues.
- **Instruction Level Parallelism** Instruction-level Parallelism: Concepts and Challenges, loop unrolling, VLIW and superscalar processors, Basic Compiler Techniques for Exposing ILP, Reducing Branch Costs with Branch Prediction, Dynamic Scheduling, Advanced Techniques for Instruction Delivery and Speculation, Limitations of ILP, Multithreading: Exploiting Thread-Level Parallelism to Improve Uniprocessor Throughput, Case Study: Dynamic Scheduling in Intel Core i7 and ARM Cortex-A8.
- **Memory Hierarchy** Introduction, Design of Memory Hierarchy, Optimizations of Cache Performance, Memory Technology and Optimizations, Virtual Memory and Virtual Machines, Case Study: Memory Hierarchies in Intel Core i7 and ARM Cortex-A8.
- **Thread Level Parallelism** - Introduction, Shared-Memory Multicore Systems, Performance Metrics for Shared-Memory Multicore Systems, Cache Coherence Protocols, cache coherence on snoopy buses, Scalable multiprocessors, Directory-based cache coherence, Synchronization, Memory Consistency, Multithreaded Programming using OpenMP, Case Study: Intel Skylake and IBM Power8.
- **Data Level Parallelism** Introduction, Vector Architecture, SIMD Instruction Set Extensions for Multimedia, Graphics Processing Units, GPU Memory Hierarchy, Detecting and Enhancing Loop- Level Parallelism, CUDA Programming, Case Study: Nvidia Maxwell.

Laboratory:

- Designing parallel version of basic algorithms,
- Multithreaded Programming using OpenMP,
- CPU-GPU programming, CUDA Programming

Text Books:

1. J.L. Hennessy and D.A. Patterson. Computer Architecture: A Quantitative Approach. 5th Edition, Morgan Kauffmann Publishers, 2012.

References:

1. J.P. Shen and M.H. Lipasti. Modern Processor Design: Fundamentals of Superscalar Processors. McGraw-Hill Publishers, 2005.
2. K. Huang, Naresh Jotwani, Advanced Computer Architecture: Parallelism, Scalability, Programmability, 3e, 2015,
3. V. Kumar et el. An Introduction to Parallel Computing: Design and Analysis of Algorithms, 2e, Pearson, 2004.
4. D.B. Kirk and W.W. Hwu. Programming Massively Parallel Processors. 2e, Morgan Kauffmann Publishers, 2012.
5. OpenMP programming (https://www.openmp.org)
6. CUDA programming (https://developer.nvidia.com/cuda-zone)

Theory of Computation CO422

3-1- 0: 4 Credits : 4 Hours *Prerequisites:* CO105, CO216

Abstract:

When we engage in solving any problem, the first two questions that comes to our mind:

- Is it possible using a given machine? (*computability*)
- How much time and space are required to solve it? (*complexity*)

The course CO422 explores these questions about the limits of abstract models of computing machines and reasoning about what they can and cannot compute efficiently. This course provides an exciting introduction to some of the fundamental ideas of theoretical computer science. It attempts to present a vision of "computer science beyond computers": that is, CS as a set of mathematical tools for understanding complex systems such as universes and minds.

Course Outcomes:

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

- Model, compare and analyse different computational models.
- discuss key notions of computation, such as algorithm, computability, decidability, reducibility, and complexity, through problem solving.
- Construct algorithms for different problems and argue formally about correctness on different restricted machine models of computation.

- explain the models of computation, including formal languages, grammars and automata, and their connections.
- Identify limitations of some computational models and possible methods of proving them.
- to introspect how the theoretical study in this course is applicable to various engineering applications including compiler design. artificial intelligence

Course Contents:

- Review of formal languages, grammar, and automata theory.
- Turing machines, Church-Turing thesis, Properties of Recursive & Recursively Enumerable Languages, Turing computable functions, Random Access Machines
- Halting problem, Decidability, Reducibility, Recursion theorem, Godel's incompleteness theorem, Universal TM, Rice's Theorem, Post's Correspondence Problem;
- Time Complexity: Complexity classes P, NP, and NP-completeness, the Cook-Levin Theorem, P versus NP problem and why it's hard
- Space Complexity: Savitch's Theorem, PSPACE, NL, and EXP.
- Intractability, Time and Space Hierarchy theorems.
  Introduction to emerging models of computation: Quantum computing; Probabilistic and randomized computation etc.

Text Books:

1. Michael Sipser, Introduction to the Theory of Computation Cengage, 2014
2. Lewis & Papadimitriou, Elements of the Theory of Computation, Pearson Education.

Reference Books:

1. Moore, Cristopher, and Stephan Mertens. The Nature of Computation. Oxford University Press, 2011
2. Arora, Sanjeev, and Boaz Barak. Computational Complexity: A Modern Approach. Cambridge University Press, 2009
3. Apostolos Doxiadis, Christos Papadimitriou and Alecos Papadatos, Logicomix: An epic search for truth, Bloomsbury, 2009

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:

(i) Getting students to appreciate the foundations of Artificial Intelligence and its future trends
(ii) Understanding of the basic elements constituting problems in an intelligent system and various approaches of solving them
(iii) Understanding the notions of uncertainty and decision making in real world problems
(iv) 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

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

Industrial Summer Training CO404

0-0-1: 1 Credit: 2 Hours *Prerequisites:* - CO311

Course objectives:

- To expose students to the 'real' working environment outside the academic life of the university and get acquainted with the organization structure, business operations and administrative functions.
- To have hands-on experience in software development/research so that they can relate and reinforce what has been taught at the university.
- To promote cooperation and to develop synergetic collaboration between industry and the university in promoting a knowledgeable society.
- To set the stage for future recruitment by potential employers.

Training will be of 12 weeks duration carried out during the summer break after the 6^{th}semester. The students will submit their reports in the 7^{th} semester.

Course Outcomes:

After completion of the training, the student will:

- gain valuable practical experience,
- able to define their own career interests.
- develop the students' job-related skills.
- enhance the students' computer science engineering knowledge acquired in class through field experience.
- be able to deal with the societal problems outside the university.

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.

Fundamentals of Speech Processing — CO4XX

3 - 0 - 1: 4 Credits: 5 Hours — *Prerequisites:* Signals and Systems (EL204)

Abstract:
This course is an introduction to Speech Processing technology and presents a comprehensive overview of digital speech processing that ranges from the basic nature of a speech signal through a variety of methods of representing speech in digital form, to applications in voice communication and automatic synthesis and recognition of speech. It provides a useful introduction to the wide range of important concepts such as speech signal representation, feature extraction, speech models, etc., that comprise the field of digital speech processing. The course also presents a brief overview of various speech applications areas such as speech recognition, speaker identification, speech synthesis, speech enhancements and recent topics of research in speech technology.

Course Outcomes:

Towards the end of the course the student would …

-be able to understand the mechanism of human speech production system and clearly describe how speech sounds are produced.

-be well conversant with the basic techniques used in the processing of a speech signal.

-be well conversant with a range of commonly used feature extraction techniques as well as speech modeling techniques.

-be able to use Matlab tools to analyse and model speech data so as to design speech systems.
-be able to design and develop simple speech recognition, speaker identification, speech synthesis systems.
-be motivated to explore research areas in speech processing technology.

Course Contents:

- **Introduction to Speech Processing:** Speech production and perception, linguistic aspect of speech, acoustic and articulatory phonetics, IPA, nature of speech, models for speech analysis and perception.

- **Analysis of speech:** Sampling, quantization, aliasing effects, filtering, frequency and time domain based methods, FFT, computation of pitch, spectrograms, formant analysis, LP analysis, cepstrum analysis.

- **Modeling techniques for developing speech systems:** Vector Quantization (VQ), Hidden Markov Models (HMM), Gaussian Mixture Models (GMM), Support Vector Machines (SVM), Artificial Neural Networks (ANN) and Deep Neural Networks (DNN).

- **Speech Systems:** *Speech Recognition:* Issues in speech recognition, isolated/connected word recognition, continuous speech recognition, large vocabulary continuous speech recognition. *Speech Synthesis:* Issues in speech synthesis, types of speech synthesis: Unit-selection, Statistical-parametric and Deep-learning based TTS systems. *Speaker Recognition:* Issues in speaker recognition, speaker verification vs identification, text-dependent vs text-independent speaker recognition, development of speaker recognition systems.

Practical Topics:

(i) Introduction to speech analysis tools
(ii) Speech signal to symbol transformation (transcription)
(iii) Study of speech production mechanism
(iv) Extraction of pitch and formant frequencies from the speech signal
(v) Study of quantization and aliasing effects
(vi) Formant analysis of vowels
(vii) Synthesis of vowels such as /a/, /i/ and /e/
(viii) Study issues of short term spectral analysis of speech signals
(ix) Linear prediction analysis of speech
(x) Speech classification problem.

Text Books:

1. L.R. Rabiner and R.W. Schafer, "Digital Processing of Speech Signals", Pearson Education, Delhi, India, 2004.
2. L. R. Rabiner, B. H. Jhuang and B. Yegnanarayana, "Fundamentals of speech recognition", Pearson Education, 2009.
3. D. O'Shaughnessy, "Speech Communication: Human and Machine", 2nd edition, IEEE Press, NY, USA, 1999.

Reference Books:

1. T. F. Quatieri, "Discrete time processing of speech signals", Pearson Education, 2005.
2. J. Benesty, M. M. Sondhi and Y. Huang, "Springer Handbook on Speech Processing", Springer publishers, 2008.

3. X. Huang, A. Acero and H. W. Hon, R foreword by Reddy , "Spoken Language Processing: A Guide to Theory, Algorithm and System Development", Prentice-Hall PTR, 2001.
4. S. Furui and M. Sondhi, "Advances in speech signal processing", CRC Press, 1991.
5. S. Sen, A. Dutta and N. Dey, "Audio Processing and Speech Recognition: Concepts, Techniques and Research Overviews", Springer, 2019.
6. L.Mary, "Extraction of Prosody for Automatic Speaker, Language, Emotion and Speech Recognition", Springer, 2019.
7. Ian Mcloughlin, "Applied Speech and Audio Processing, With MATLAB Examples", Cambridge University Press, 2009.

Virtual and Augmented Reality CO517

3-0-1: 4 Credits : 5 Hours *Prerequisites:* CO303(CG)

Abstract:
This course is aimed at providing an overview of the various aspects and applications of Virtual and Augmented Reality systems. It covers 3D Computer Graphics and basics concepts of Computer Vision as essential tools in the development of Virtual Reality Systems. The course discusses related technologies necessary in the Virtual and Augmented Reality systems and their evaluation.

Course Outcomes:

After completion of course, students would appreciate and be conversant with:

- Concepts of the topics in VR and AR systems design involving the foundations of VR and AR systems and the current trends
- Understanding of the basic components and peripheral systems in VR and AR System
- Understanding of the various stages in the VR and AR System pipeline
- Knowledge of practical and developmental aspects of VR applications

Course Contents:

- Unit 1: Introduction to Virtual and Augmented Reality
  - Review of Computer Graphics and Vision in VR
- Unit 2: AR/VR Peripheral and Programming foundation
  - Introduction to basic architecture and functioning of head-mounted device and other peripherals
  - Art of AR/VR programming using OpenGL/C++/Unity
- Unit 3: 3D Computer Graphics[LAV]
  - Geometry and Transformations
  - Perception and Rendering
  - Motion in Real and Virtual worlds
  - Tracking technology.
- Unit 4: Virtual Reality Systems [LAV]

    - Locomotion
    - Manipulation
    - Interaction
    - Cinematic VR, Spatial Sound
- Unit 5: Augmented Reality Systems [SCH]
    - Registration
    - Coherence
    - Visualization
- Unit 6: Evaluation of VR/AR Systems

Laboratory Work:

Laboratory work will involve

- Assignments on 3D Graphics
- Mini projects on AR/VR applications for desktop, hand-held, and head-worn displays

Applications can be deployed on handheld Android and iOS devices with cameras (smartphones and tablets) or head-worn displays.

Text Books and References:

- LaValle "Virtual Reality", Cambridge University Press, 2016**[LAV]**
- D. Schmalstieg and T. Höllerer. Augmented Reality: Principles and Practice. Addison-Wesley, Boston, 2016, ISBN-13 978-0-32-188357-5**[SCH]**

Reference Books

- J. LaViola Jr., E. Kruijff, R. McMahan, D. Bowman, and I. Poupyrev. 3D User Interfaces: Theory and Practice, 2nd Edition. Addison-Wesley, Boston, 2017, ISBN-13 978-0-13-403432-4 **[3DUI]**
- Marschner, Shirley "Fundamentals of Computer Graphics", 4th Edition, CRC Press 2016 **[MAR]**