



خواجہ معین الدین چشتی اردو، عربی-فارسی یونیورسٹی، لکھنؤ، اتر پردیش، انڈیا  
ख्वाजा मुईनुद्दीन चिश्ती उर्दू, अरबी-फ़ारसी विश्वविद्यालय, लखनऊ, उत्तर प्रदेश, भारत  
Khwaja Moinuddin Chishti Urdu, Arabi-Farsi University, Lucknow, Uttar Pradesh, India  
U.P. STATE GOVERNMENT UNIVERSITY  
(Recognised U/S 2(f) & 12 (B) of the UGC Act 1956 & B.Tech. approved by AICTE)

## FACULTY OF ENGINEERING & TECHNOLOGY

COMPUTER SCIENCE & ENGINEERING  
with Specialization using AI & ML



### Curriculum Structure

(Third Year- V Semester)

[Effective from Session 2021-22]

## STUDY & EVALUATION SCHEME

**B.Tech. (CSE specialization with AI&ML)**

**III Year: V Semester**

S.No.	Subject code	Subject name	L	T	P	Sessional Assessment			SEE	Subject Total	Credit
						MST	TA	Total			
<b>THEORY SUBJECT</b>											
1	ACS501	Design & Analysis of Algorithms	3	0	0	15	15	30	70	100	3
2	ACS502	Python Programming	3	0	0	15	15	30	70	100	3
3	ACS503	Software Engineering	3	0	0	15	15	30	70	100	3
4	ACS504	Embedded Systems	3	1	0	15	15	30	70	100	4
5	ACS505	Formal Language & Automata Theory	3	1	0	15	15	30	70	100	4
6	EC501	Microprocessor	3	0	0	15	15	30	70	100	3
7	GP501	General Proficiency	-	-	-	-	-	50	0	50	0
<b>PRACTICAL/DESIGN/DRAWING</b>											
7	ACS551	Design & Analysis of Algorithms Lab	0	0	2	15	15	30	70	100	1
8	ACS552	Python Programming Lab	0	0	2	15	15	30	70	100	1
9	ACS553	Software Engineering Lab	0	0	2	15	15	30	70	100	1
10	ACS554	Embedded Systems Lab	0	0	2	15	15	30	70	100	1
		<b>Total</b>	<b>15</b>	<b>2</b>	<b>6</b>					<b>1000</b>	<b>24</b>

L- Lecture

T -Tutorial

P-Practical

MST- Mid Semester Test

TA-Teacher's Assessment

SEE- Semester End Examination



**DESIGN & ANALYSIS OF ALGORITHMS  
(ACS501)**

<b>Objective:</b> To understand the importance of algorithm and its complexity of an algorithm in terms of time and space complexities.	
Unit	Topic
<b>I</b>	<b>Introduction:</b> Algorithms, Analyzing algorithms, Complexity of algorithms, Growth of Functions, Recurrences, Substitution method, Iteration method, Master method, Merge Sort, Quick-Sort, Heap Sort, Shell Sort, Sorting in linear time.
<b>II</b>	<b>Advanced Data Structures:</b> Red-black trees, Augmenting data structures, Order-statistic tree, B-Trees, Binomial heaps, Fibonacci heaps.
<b>III</b>	<b>Dynamic Programming:</b> Elements of dynamic programming, Assembly-line scheduling problem, Matrix chain multiplication, finding longest common subsequence, 0/1 Knapsack problem; <b>Greedy Algorithm:</b> Elements of greedy strategy, Activity selection problem, Huffman encoding, Task-scheduling problem, Knapsack problem, Amortized analysis.
<b>IV</b>	<b>Graph Algorithms:</b> Searching in graph, Spanning trees, Minimum cost spanning trees: Kruskal's and Prim's algorithms; Single source shortest path algorithms, Dijkstra's and Bellman Ford algorithms; All pair shortest paths algorithms, Floyd Warshal's algorithm, Network flow problem. Backtracking, Graph Coloring, n-Queen Problem, Hamiltonian Cycles and Sum of Subsets, Branch and Bound with Examples Such as Travelling Salesman Problem.
<b>V</b>	<b>String Matching Algorithms:</b> Naïve string-matching algorithm, Rabin-Karp algorithm, Knuth-Morris-Pratt algorithm. Introduction of NP-completeness, Randomized algorithms and Approximation Algorithms

**Text Book (s):**

1. Introduction to Algorithms, Thomas H Cormen, Charles E Lieserson, Ronald L Rivest and Clifford Stein, MIT Press/McGraw-Hill.
2. Fundamentals of Algorithms – E. Horowitz et al.
3. Design & Analysis of Algorithms, S. Sridhar, Oxford
4. Design & Analysis of Algorithms, Sharma, Khanna Publishing House, N.Delhi



## Python Programming (ACS502)

**Objective:** To familiarize the students with advanced databases and techniques of retrieving and storing information.

Unit	Topic
<b>I</b>	<b>Introduction To Python:</b> Installation and Working with Python Understanding Python variables Python basic Operators Understanding python blocks <b>Values and Variables :</b> Integer and String Values, Identifiers, User Input, String Formatting, Expressions and Arithmetic Examples
<b>II</b>	<b>Python Data Types:</b> Declaring and using Numeric data types: int, float, complex Using string data type and string operations Defining list and list slicing Use of Tuple data type
<b>III</b>	<b>Python Conditional Statements and looping:</b> If, If- else, Nested if-else For, While Nested loops
<b>IV</b>	<b>Python String, List And Dictionary Manipulations:</b> Building blocks of python programs, Understanding string in build methods, List manipulation using in build methods ,Dictionary manipulation Programming using string, list and dictionary in build functions
<b>V</b>	<b>Python Object Oriented Programming:</b> Oops Concept of class, object and instances Constructor, class attributes and destructors ,Real time use of class in live projects ,Inheritance , overlapping and overloading operators Adding and retrieving dynamic attributes of classes Programming using Oops support..

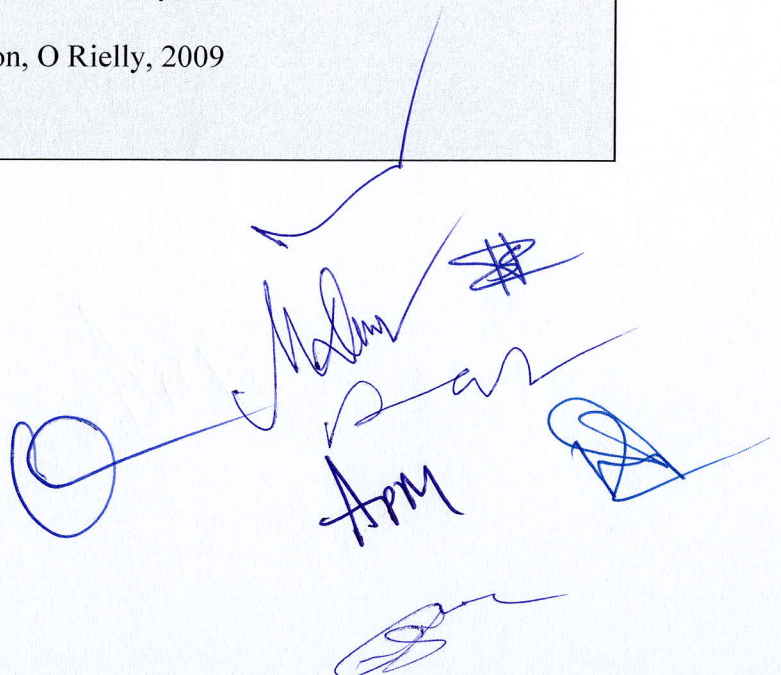
### References:

Chun, J Wesley, Core Python Programming, Second Edition, Pearson, 2007 Reprint 2010

#### Essential Reading / Recommended Reading

[1] Barry, Paul, Head First Python, 2nd Edition, O Rielly, 2010

[2] Lutz, Mark, Learning Python, 4th Edition, O Rielly, 2009





## SOFTWARE ENGINEERING (ACS503)

**Objective:** The course is aimed at enhancing skills that will enable the student to develop business software's that are simple reliable and capable of modification as per requirement.

Unit	Topic
<b>I</b>	Introduction to Software Engineering, Software Components, Software Characteristics, Software Crisis, Software Engineering Processes. Software Development Life Cycle (SDLC) Models: Water Fall Model, Prototype Model, Spiral Model, Evolutionary Development Models, Iterative Enhancement Models.
<b>II</b>	Software Requirement Specifications (SRS). Requirement Engineering Process: Elicitation, Analysis, Documentation, Review and Management of User Needs, Feasibility Study, Information Modeling, Data Flow Diagrams, Entity Relationship Diagrams, Decision Tables, SRS Document, IEEE Standards for SRS. Software Quality Attributes, Software Quality Assurance (SQA): Verification and Validation, SQA Plans, Software Quality Frameworks, ISO 9000 Models, SEI-CMM Model.
<b>III</b>	Software Design: Basic Concept of Software Design, Architectural Design, Low Level Design: Modularization, Design Structure Charts, Pseudo Codes, Flow Charts, Coupling and Cohesion Measures, Design Strategies: Function Oriented Design, Object Oriented Design, Top-Down and Bottom-Up Design. Software Measurement and Metrics: Various Size Oriented Measures: Halstead's Software Science, Function Point (FP) Based Measures, Cyclomatic Complexity Measures: Control Flow Graphs.
<b>IV</b>	Software Testing: Testing Objectives, Unit Testing, Integration Testing, Acceptance Testing, Regression Testing, Testing for Functionality and Testing for Performance, Top-Down and Bottom-Up Testing Strategies: Test Drivers and Test Stubs, Structural Testing (White Box Testing), Functional Testing (Black Box Testing), Test Data Suit Preparation, Alpha and Beta Testing of Products. Static Testing Strategies: Formal Technical Reviews (Peer Reviews), Walk Through, Code Inspection, Compliance with Design and Coding Standards.
<b>V</b>	Software Maintenance and Software Project Management, Software as an Evolutionary Entity, Need for Maintenance, Categories of Maintenance: Preventive, Corrective and Perfective Maintenance, Cost of Maintenance, Software Re-Engineering, Reverse Engineering. Software Configuration Management Activities, Change Control Process, Software Version Control, An Overview of CASE Tools. Estimation of Various Parameters such as Cost, Efforts, Schedule/Duration, Constructive Cost Models (COCOMO), Resource Allocation Models, Software Risk Analysis and Management.

### References:

1. Software Engineering: A Practitioner's Approach, Pressman Roger, TMH, 2009.
2. An Integrated Approach to Software Engineering, Pankaj Jalote. Narosa Pub, 2014.
3. Software Engineering Concepts: Richard Fairly, Tata McGraw Hill, 2015.

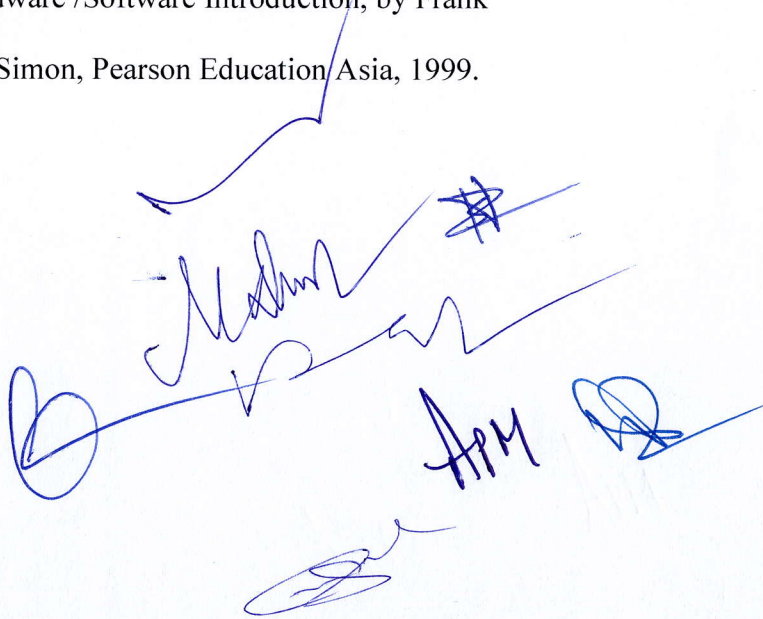


## EMBEDDED SYSTEM (ACS504)

<b>Objective:</b> To introduce the basic concepts of Embedded Systems and the various techniques used for Embedded Systems with real time examples.	
Unit	Topic
<b>I</b>	Hardware Concepts -Application and characteristics of embedded systems, Overview of Processors and hardware Units in an embedded system, General purpose processors, Microcontrollers: 8051.
<b>II</b>	Application- Specific Circuits (ASICs), ASIP, FPGA, ARM-based System on a Chip (SoC), Network on Chip (NoC), Levels of hardware modelling, Verilog, Sensors, A/D-D/A converters, Actuators, Interfacing using RS-232, UART, USB, I2C, CAN bus, Flexray, SRAM and DRAM, Flash memory.
<b>III</b>	Real-Time Operating Systems- Real-Time Task Scheduling: Some important concepts, Types of real-time tasks and their characteristics, Task scheduling, Clock-Driven scheduling, Hybrid schedulers, Event-Driven scheduling, Earliest Deadline First (EDF) scheduling, Rate monotonic algorithm (RMA).
<b>IV</b>	Commercial Real-time operating systems: Time services, Features of a Real-time operating system, Unix-based Real-time operating systems, POSIX-RT, A survey of contemporary Real- time operating systems, Microkernelbased systems, Benchmarking real-time systems.
<b>V</b>	Embedded Application Development - UML 2.0, State charts, General language characteristics, MISRA C, Hardware/Software Co- design, Hardware/software partitioning, Testing embedded systems, Design for testability and Self-test.

### References:

1. Embedded Systems Design – A Unified Hardware /Software Introduction, by Frank Vahid and Tony Givargis, John Wiley, 1999.
2. An Embedded Software Primer, by David E.Simon, Pearson Education/Asia, 1999.





## THEORY OF AUTOMATA & FORMAL LANGUAGE (ACS505)

**Objective:** The objective of this course is to provide basic definitions that are associated with theory of computation and to give an overview, applications, environment of computation.

Unit	Topic
<b>I</b>	<b>Introduction:</b> Alphabets, Strings and Languages; Automata and Grammars, Chomsky's classification. Finite Automata: Deterministic finite Automata (DFA)-Formal Definition, Simplified notation: State transition graph, Transition table, Language of DFA, Nondeterministic finite Automata (NFA), NFA with epsilon transition, Language of NFA, Equivalence of NFA and DFA, Minimization of Finite Automata, Distinguishing one string from other, Myhill-Nerode Theorem.
<b>II</b>	<b>Regular Expression:</b> Regular expression (RE), Definition, Operators of regular expression and their precedence, Algebraic laws for Regular expressions, Kleen's Theorem, Regular expression to FA, DFA to Regular expression, Arden Theorem, Regular Languages and Its Properties: Non Regular Languages, Pumping Lemma for regular Languages. Application of Pumping Lemma, Closure properties of Regular Languages, Decision properties of Regular Languages, FA with output: Moore and Mealy machine, Equivalence of Moore and Mealy Machine, Applications and Limitation of FA.
<b>III</b>	<b>Context free grammar (CFG) and Context Free Languages (CFL):</b> Definition, Examples, Derivation, Derivation trees, Ambiguity in Grammer, Inherent ambiguity, Ambiguous to Unambiguous CFG, Useless symbols, Simplification of CFGs, Normal forms for CFGs: CNF and GNF, Closure properties of CFLs, Decision Properties of CFLs: Emptiness, Finiteness and Memership, Pumping lemma for CFLs.
<b>IV</b>	<b>Push Down Automata (PDA):</b> Description and definition, Instantaneous Description, Language of PDA, Acceptance by Final state, Acceptance by empty stack, Deterministic PDA, Equivalence of PDA and CFG, CFG to PDA and PDA to CFG, Two stack PDA.
<b>V</b>	<b>Turing machines (TM):</b> Basic model, definition and representation, Instantaneous Description, Language acceptance by TM, Variants of Turing Machine, TM as Computer of Integer functions, Universal TM, Church's Thesis. Recursive and Recursively Enumerable languages. Undecidability: Halting problem, Introduction to Undecidability, Undecidable problems about TMs. Post correspondence problem (PCP), Modified PCP, Introduction to recursive function theory.

### References:

1. J Hopcroft, JD Ullman, R Motwani, Introduction to Automata Theory, Languages and Computation, Pearson, 2006
2. M Sipser, Introduction to the Theory of Computation, Thomson, 2006.



**MICROPROCESSOR  
(EC501)**

<b>UNIT I</b> Introduction to 8085A CPU architecture-register organization, addressing modes and their features. Software instruction set and Assembly Language Programming. Pin description and features
<b>UNIT II</b> Instruction cycle, machine cycle, Timing diagram. Hardware Interfacing: Interfacing memory, peripheral chips (IO mapped IO & Memory mapped IO).
<b>UNIT III</b> Interrupts and DMA. Peripherals: 8279, 8255, 8251, 8253, 8237, 8259, A/D and D/A converters and interfacing of the same.
<b>UNIT IV</b> 16 bit processors: 8086 and architecture, segmented memory has cycles, read/write cycle in min/max mode. Reset operation, wait state, Halt state, Hold state, Lock operation, interrupt processing. Addressing modes and their features. Software instruction set (including specific instructions like string instructions, repeat, segment override, lock prefizers and their use) and Assembly Language programming with the same
<b>UNIT V</b> Typical applications of a microprocessor. Brief overview of some other microprocessors (eg. 6800 Microprocessor).

**Reference Books:**

1. Microprocessor architecture, programming and applications with 8085/8085A, Wiley eastern Ltd, 1989 by Ramesh S. Gaonkar.
2. Intel Corp: The 8085 / 8085A. Microprocessor Book – Intel marketing communication, Wiley inter science publications, 1980.
3. An introduction to micro computers Vol. 2 – some real Microprocessor – Galgotia Book Source, New Delhi by Adam Osborne and J. Kane
4. Advanced Microprocessors by Ray and Bhurchandi - TMH
5. Intel Corp. Micro Controller Handbook – Intel Publications, 1994.
6. Microprocessors and Interfacing by Douglas V. Hall, McGraw Hill International Ed. 1992
7. Assembly Language Programming the IBM PC by Alan R. Miller, Subex Inc, 1987
8. The Intel Microprocessors: 8086/8088, 80186, 80286, 80386 & 80486, Bary B. Brey, Prentice Hall, India 1996

Handwritten signatures and initials in blue ink, including a large signature that appears to be 'J. M.', a signature that looks like 'R. M.', and other initials like 'APM' and 'S. R.'.



**DESIGN & ANALYSIS OF ALGORITHM LAB**  
**(ACS551)**

**LIST OF EXPERIMENTS**

1. Implementation of Quick Sort and Merge Sort.
2. Implementation of Linear-time Sorting Algorithms.
3. Implementation of Red-Black Tree operations.
4. Implementation of Binomial Heap operations.
5. Implementation of an application of Dynamic Programming.
6. Implementation of an application of Greedy Algorithm.
7. Implementation of Minimum Spanning Tree Algorithm.
8. Implementation of Single-pair shortest path Algorithm.
9. Implementation of All-pair shortest path Algorithm.
10. Implementation of String Matching Algorithm.

*Handwritten signatures and initials in blue ink:*  
A large signature with a checkmark above it.  
A signature with a checkmark below it.  
The initials "APM".  
A signature with a checkmark below it.  
A signature with a checkmark below it.



**Python Programming LAB**  
(ACS552)

**LIST OF EXPERIMENTS**

1. Implement a sequential search
2. Create a calculator program
3. Explore string functions
4. Implement Selection Sort
5. Implement Stack
6. Read and write into a file
7. Demonstrate usage of basic regular expression
8. Demonstrate use of List
9. Demonstrate use of Dictionaries

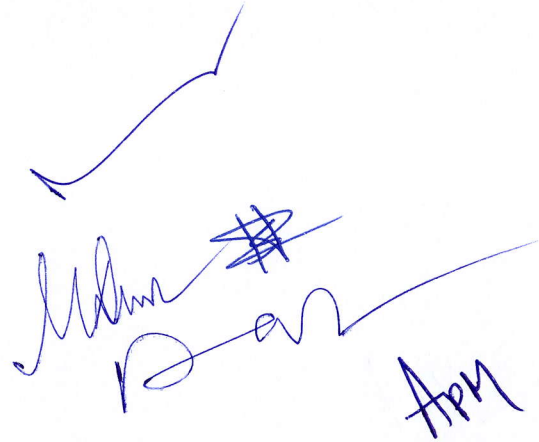
*[Handwritten signatures and initials in blue ink]*

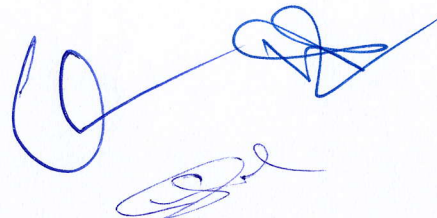


**SOFTWARE ENGINEERING LAB**  
**(ACS553)**

**LIST OF EXPERIMENTS**

1. Introduction to Microsoft Project Professional.
2. Basic steps required to create project and prepare it for data entry (project tasks, sequence the tasks and estimate task duration).
3. Setting up a project [Eating Breakfast] and establish the basic constraints that project will use for its calculation. Analyze the project from different view [Gantt Chart, Network Diagram]
4. Setting up a project [Refurbishment of Workshop] and identifying relationship among the different task and subtask.
5. Setting up a project [Exam Cell Activities] and explain how to enter resources and specific information in Microsoft Project and resources to specific tasks.
6. Case Study: Project Windows 8 (Module works on windows Vista and now transform the module to work on Window 8).

  
A handwritten signature in blue ink, possibly reading 'M. M. #', with a checkmark above it. Below the signature are the initials 'Ran' and 'APM'.


  
A handwritten signature in blue ink, possibly reading 'S. S.', with a checkmark above it.



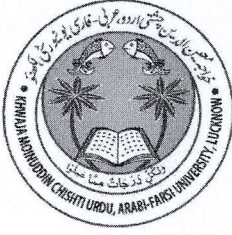
**Embedded System LAB**  
(ACS554)

**LIST OF EXPERIMENTS**

1. Write a program to toggle all the led to port and with some time delay using ARM7
2. Write a program to interface LCD with ARM7
3. Write a program to interface 4\*4 matrix keypad with ARM7
4. Write a program for interfacing LED and PWM and to verify the output in the ARM7
5. Write a program to interface Stepper motor with ARM7
6. Write a program for interfacing of DC motor with ARM7
7. Write a program to study and characteristics of the programmable gain amplifier (PGA)
8. Write a Program realization of low pass, high pass and band pass filters and their characteristics
9. Write a program to interface ADC and DAC with PSOC







خواجہ معین الدین چشتی اردو، عربی-فارسی یونیورسٹی، لکھنؤ، اتر پردیش، انڈیا  
ख्वाजा मुईनुद्दीन चिश्ती उर्दू، अरबी-फारसी विश्वविद्यालय, लखनऊ, उत्तर प्रदेश, भारत  
Khwaja Moinuddin Chishti Urdu, Arabi-Farsi University, Lucknow, Uttar Pradesh, India  
U.P. STATE GOVERNMENT UNIVERSITY  
(Recognised U/S 2(f) & 12 (B) of the UGC Act 1956 & B.Tech. approved by AICTE)

## FACULTY OF ENGINEERING & TECHNOLOGY

COMPUTER SCIENCE & ENGINEERING  
with Specialization using AI & ML



### Curriculum Structure

(Third Year- VI Semester)

[Effective from Session 2021-22]



## STUDY & EVALUATION SCHEME

**B.Tech. (CSE specialization with AI&ML)**

**III Year: VI Semester**

S.No.	Subject code	Subject name	L	T	P	Sessional			SEE	Subject Total	Credit
						MST	TA	Total			
<b>THEORY SUBJECT</b>											
1	ACS601	Computer Networks	3	0	0	15	15	30	70	100	3
2	ACS602	Compiler Design	3	1	0	15	15	30	70	100	4
3	ACS603	Neural Network & Deep Learning	3	1	0	15	15	30	70	100	4
4	ACS604	Speech and Natural Language Processing	3	0	0	15	15	30	70	100	3
5	ACS061 - 064	Elective -I	3	0	0	15	15	30	70	100	3
6	AS601	Engineering Economics	2	0	0	15	15	30	70	100	2
7	GP601	General Proficiency	-	-	-	-	-	50	0	50	0
<b>PRACTICAL/DESIGN/DRAWING</b>											
7	ACS651	Computer Networks Lab	0	0	2	15	15	30	70	100	1
8	ACS652	Compiler Design Lab	0	0	2	15	15	30	70	100	1
9	ACS653	Project - I	0	0	6	0	100	100	0	200	3
		<b>Total</b>	<b>17</b>	<b>2</b>	<b>10</b>					<b>1000</b>	<b>24</b>

**Student has to undergo a summer training of 45 days at the end of VI Sem.**

L- Lecture

T -Tutorial

P-Practical

MST- Mid Semester Test

TA-Teacher's Assessment

SEE- Semester End Examination



## COMPUTER NETWORKS (ACS601)

**Objective:** The objective of this course is to provide basic exposure to computer networks theory and implementations.

Unit	Topic
<b>I</b>	<p><b>Introduction:</b> Networks, Internet, Network Components, Network Categories, Applications of Computer Networks</p> <p><b>Reference Models:</b> Concept of Layering, OSI Model, TCP/IP Protocol Suite, Functions of Layers</p> <p><b>Physical Layer:</b> Transmission Mode, Physical Topology, Multiplexing, Transmission Media, Switching</p>
<b>II</b>	<p><b>Data Link Layer:</b> Design Issues, Error Detection and Correction Techniques, Elementary Data Link Protocols, Sliding Window Protocols, Multiple Access Protocols, Ethernet, Connecting Devices</p>
<b>III</b>	<p><b>Network Layer:</b> Logical addressing, IPv4 Addresses, NAT, IPv6 Addresses, Internet Protocol, IPv4, IPv6, Internetworking, Internet Control Protocols, Routing Algorithms, Distance Vector Routing, Link State Routing, Routing in the Internet</p>
<b>IV</b>	<p><b>Transport Layer:</b> Process-to-Process Delivery, Transport Layer Protocols, UDP, User Datagram, TCP, TCP Segment, TCP Connection, Flow Control and Error Control, TCP Transmission Policy, Principles of Congestion Control, TCP Congestion Control, Quality of Service.</p>
<b>V</b>	<p><b>Application Layer:</b> Principles of Network Applications, WWW and HTTP, Non-Persistent and Persistent Connections, Cookies, Web Caching, File Transfer, Remote Logging, Electronic Mail in the Internet, Domain Name System, Security: Introduction, Cryptography and Cryptanalysis, Public Key Cryptography Algorithms, RSA Algorithm, DES, Authentication and Authorization</p>

**References:**

1. AS Tanenbaum, DJ Wetherall, Computer Networks, Prentice-Hall, 2010.
2. LL Peterson, BS Davie, Computer Networks: A Systems Approach, Morgan-Kaufman, 2011.
3. W Stallings, Cryptography and Network Security, Principles and Practice, Prentice-Hall, 2005.



## COMILER DESIGN (ACS602)

**Objective:** Students will have a fair understanding of some standard passes in a general-purpose compiler.

Unit	Topic
<b>I</b>	<b>Introduction to Compiler:</b> Phases and passes, Bootstrapping, Finite state machines and regular expressions and their applications to lexical analysis, implementation of lexical analyzers,, LEX-compiler, Formal grammars and their application to syntax analysis,, ambiguity, The syntactic specification of programming languages: Context free grammars, derivation and parse trees, capabilitiesofCFG.
<b>II</b>	<b>Basic Parsing Techniques:</b> Parsers, Shift reduce parsing, operator precedence parsing, top down parsing, predictive parsers Automatic Construction of efficient Parsers: LR parsers, the canonical Collection of LR(0) items, constructing SLR parsing tables, constructing Canonical LR parsing tables, Constructing LALR parsing tables, using ambiguous grammars, an automatic parser generator, YACC tool.
<b>III</b>	<b>Syntax-directed Translation:</b> Syntax-directed Translation schemes, Intermediate code, postfix notation, Parse trees & syntax trees, three address code, quadruple & triples, Translation of simple statements and control flow statements, Type checking, Type conversions, Equivalence of type expressions, Overloading of functions and operations.
<b>IV</b>	<b>Symbol Tables:</b> Data structure for symbols tables, representing scope information. <b>Run-TimeAdministration:</b> Implementation of simple stack allocation scheme, storage allocation in block structured language. Error Detection & Recovery: Lexical Phase errors, syntactic phase errors semantic errors.
<b>V</b>	<b>Code Generation:</b> Design Issues, the Target Language. Addresses in the Target Code, Basic Blocks and Flow Graphs, Optimization of Basic Blocks, Code Generator. Code optimization: Machine-Independent Optimizations, Loop optimization, DAG representation of basic blocks, value numbers and algebraic laws, Global Data-Flow analysis.

### References:

1. K.D. Cooper, and Linda Torczon, Engineering a Compiler, Morgan Kaufmann, 2011.
2. K.C. Louden, Compiler Construction: Principles and Practice, Cengage Learning, 1997.
3. D. Brown, J. Levine, and T. Mason, LEX and YACC, OReilly Media, 1992.

Handwritten signatures and initials in blue ink, including a large signature on the left, initials 'APM' in the center, and several other scribbles on the right.



## NEURAL NETWORK AND DEEP LEARNING (ACS603)

**Objective:** To teach fundamentals of neuro computing with applications to computer engineering problems.

Unit	Topic
<b>I</b>	Introduction: Neural Network, Human Brain, Biological and Artificial Neurons, Model of Neuron Knowledge Representation, Artificial Intelligence and Neural Network, Network Architecture, Basic Approach of the working of ANN- Training, Learning and Generalization.
<b>II</b>	Supervised Learning: Single Layer Networks, Perception- Linear Separability, Limitations of Multi Layer Network Architecture, Back Propagation Algorithm (BPA) and Other Training Algorithms.
<b>III</b>	Application of Adaptive Multi- Layer Network Architecture, Recurrent Network, Feed-Forward Networks Radial-Basic-Function (RBF) Networks.
<b>IV</b>	Unsupervised Learning: Winner- Task-All Networks, Hamming Networks, Maxnet, Simple Competitive Learning Vector- Quantization, Counter-Propoagation Network, Adaptive Resonance Theory, Kohonen's Self Organizing Maps, Principal Component Analysis.
<b>V</b>	Introduction To Deep Learning, Deep Learning Models, Restricted Boltzmann Machines, Deep Belief Nets, Convolution Networks, Recurrent Nets, Deep Learning Platforms

### References:

1. Simon Haykin, "Neural Network – A Comprehensive Foundation", Macmillan Pub.1994,
2. K.Mahrotra, C.K. Mohan and Sanjay Ranka, Elements of Artifical Neural Network, MIT Press, 1997.
3. J.M. Zurada, "Introduction to Artificial Neural network", Jaico Publihers, 2012.
4. Limin Fu. "Neural Networks in Computer Intelligence", TMH, 2005.



Handwritten signatures and initials in blue ink, including a large signature on the left, a signature with a hash symbol (#) in the middle, and several other initials and signatures on the right and bottom.



**SPEECH AND NATURAL LANGUAGE PROCESSING**  
**(ACS604)**

**Objective:** To teach how to design, code, debug and document programs using techniques of good programming style.

Unit	Topic
I	Introduction to Natural Language Understanding: The study of Language, Applications of NLP, Evaluating Language Understanding Systems, Different levels of Language Analysis, Representations and Understanding, Organization of Natural language Understanding Systems, Linguistic Background: An outline of English syntax.
II	Introduction to semantics and knowledge representation, Some applications like machine translation, database interface.
III	Grammars and Parsing: Grammars and sentence Structure, Top-Down and Bottom-Up Parsers, Transition Network Grammars, Top-Down Chart Parsing. Feature Systems and Augmented Grammars: Basic Feature system for English, Morphological Analysis and the Lexicon, Parsing with Features, Augmented Transition Networks.
IV	Grammars for Natural Language: Auxiliary Verbs and Verb Phrases, Movement Phenomenon in Language, Handling questions in Context-Free Grammars. Human preferences in Parsing, Encoding uncertainty, Deterministic Parser.
V	Ambiguity Resolution: Statistical Methods, Probabilistic Language Processing, Estimating Probabilities, Part-of-Speech tagging, Obtaining Lexical Probabilities, Probabilistic Context-Free Grammars, Best First Parsing. Semantics and Logical Form, Word senses and Ambiguity, Encoding Ambiguity in Logical Form.

**References:**

1. Akshar Bharti, Vineet Chaitanya and Rajeev Sangal "NLP: A Paninian Perspective", PHI, 1994.
2. James Allen "Natural Language Understanding" Pearson, 1994 .

Handwritten signatures and initials in blue ink, including a large signature on the left, a crossed-out signature in the middle, and several other initials and marks on the right and bottom.

**COMPUTER NETWORKS LAB**  
**(ACS651)**

**LIST OF EXPERIMENTS**

1. To learn basics of the packet tracer simulator tool.
2. Write a program in C to implement bit stuffing and character stuffing.
3. To connect the computers in Local Area Network and to detect collision of packets.
4. To configure DHCP and DNS server for a given network in packet tracer simulator tool.
5. Write a C program to get the MAC or Physical address of the system using ARP (Address Resolution Protocol) and to subnet a given network according to the requirements in packet tracer simulator tool. .
6. To configure router using command line. Also observe the datagram formats in packet tracer simulator tool.
7. To configure NAT for a given network in packet tracer simulator tool.
8. Write a program to implement TCP & UDP Sockets.
9. Write a C program to transmit a character, a string and a file from one computer to another using RS-232 cable and to configure static routing in packet tracer simulator tool.
10. To configure dynamic routing protocols in packet tracer simulator tool.





**COMILER DESIGN LAB**  
**(ACS652)**

**LIST OF EXPERIMENTS**

1. Write a program to check whether a string belongs to the grammar or not.
2. Practice of Lex of Compiler writing.
3. Write a LEX program to count number of printf and scanf from a given c program file and replace them with write and read respectively.
4. Write a program to check whether a grammar is left recursive and remove left recursion.
5. Write a program to remove left factoring
6. Write a program to compute FIRST and FOLLOW of non-terminals.
7. Write a program to check whether a grammar is Operator precedent. .
8. Practice of Yacc of Compiler writing.
9. Write a YACC program to recognize the grammer[  $a^n b^n > 0$  ] . Test whether the following string belongs to this grammer.
10. Write a YACC & LEX program to identify valid if and if-else statement.

  
A collection of handwritten signatures in blue ink, including names like 'Mahn', 'Ran', 'Amy', and others, some with checkmarks or initials.

## Project -I (ACS653)

The object of *Project Work I* is to enable the student to take up investigative study in the broad field of *Computer Science & Engineering*, either fully theoretical/practical or involving both theoretical and practical work to be assigned by the Department on an individual basis or two/three students in a group, under the guidance of a Supervisor. This is expected to provide a good initiation for the student(s) in R&D work. The assignment to normally include:

- 1) Survey and study of published literature on the assigned topic;
- 2) Working out a preliminary Approach to the Problem relating to the assigned topic;
- 3) Conducting preliminary
- 4) Analysis/Modeling/Simulation/Experiment/Design/Feasibility;
- 5) Preparing a Written Report on the Study conducted for presentation to the
- 6) Department;
- 7) Final Seminar, as oral Presentation before a Departmental Committee.



Handwritten signatures in blue ink, including a large signature on the left, a signature with a crossed-out word above it, and several other signatures on the right and bottom.