DESIGN & ANALYSIS OF ALGORITHM
(CS501)

Objective: To understand the importance of algorithm and its complexity of an algorithm in terms of time and space complexities.	
Unit	**Topic**
I	**Introduction**: 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.
II	**Advanced Data Structures:** Red-black trees, Augmenting data structures, Order-statistic tree, B-Trees, Binomial heaps, Fibonacci heaps.
III	**Dynamic Programming:** Elements of dynamic programming, Assembly-line scheduling problem, Matrix chain multiplication, finding longest common subsequence,0/1 Knapsack problem; **Greedy Algorithm:** Elements of greedy strategy, Activity selection problem, Huffman encoding, Task-scheduling problem, Knapsack problem, Amortized analysis.
IV	**Graph Algorithms:** 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.
V	**String Matching Algorithms:** Naïve string-matching algorithm, Rabin-Karp algorithm, Knuth-Morris-Pratt algorithm. Introduction of NP-completeness, Randomized algorithms and Approximation Algorithms

References:

1. "Introduction to Algorithms" by Cormen, Leiserson, Rivest, Stein, MIT Press, 2002.
2. "The Design and Analysis of Computer Algorithms" by Aho, Hopcroft, Ullman, Pearson, 2007.
3. "Algorithm Design" by Kleinberg and Tardos, Pearson, 2005.

JAVA PROGRAMMING
(CS502)

Objective: The goal of subject is to provide the students a broad exposure to the advance object oriented programming field in order to be prepared for follow-on study.	
Unit	**Topic**
I	Java Basics History of Java, Java buzzwords, datatypes, variables, scope and life time of variables, arrays, operators, expressions, control statements, type conversion and costing, simple java program, classes and objects – concepts of classes, objects, constructors, methods, access control, this keyword, garbage collection, overloading methods and constructors, parameter passing, recursion, string handling.
II	Packages and Interfaces : Defining, Creating and Accessing a Package, Understanding CLASSPATH, importing packages, differences between classes and interfaces, defining an interface, implementing interface, applying interfaces, variables in interface and extending interfaces. Exploring packages – Java.io, java.util. Exception handling and multithreading – Concepts of exception handling, benefits of exception handling, Termination or resumptive models, exception hierarchy, usage of try, catch, throw, throws and finally, built in exceptions, creating own exception sub classes. Differences between multi threading and multitasking, thread life cycle, creating threads, synchronizing threads, daemon threads, thread groups.
III	Event Handling : Events, Event sources, Event classes, Event Listeners, Delegation event model, handling mouse and keyboard events, Adapter classes, inner classes.The AWT class hierarchy, user interface components-labels, button, canvas, scrollbars, text components, check box, check box groups, choices, lists panels – scrollpane, dialogs, menubar, graphics, layout manager – layout manager types – boarder, grid, flow, card and grib bag.
IV	Applets – Concepts of Applets, differences between applets and applications, life cycle of an applet, types of applets, creating applets, passing parameters to applets. Swing – Introduction, limitations of AWT, MVC architecture, components, containers, exploring swing- JApplet, JFrame and JComponent, Icons and Labels, text fields, buttons – The JButton class, Check boxes, Radio buttons, Combo boxes, Tabbed Panes, Scroll Panes, Trees, and Tables.
V	Networking – Basics of network programming, addresses, ports, sockets, simple client server program, multiple clients, Java .net package, Web Servers and Servlets: Tomcat web server, Introduction to Servelets: Lifecycle of a Serverlet, Web Server operations, general server characteristics, Security Issues, Structure of Web Application, Deploying Web Application, Introduction to Model View Controller (MVC) Architecture, its structure, components.

References:

1. Java; the complete reference, Herbert schildt, TMH, 2011
2. Understanding OOP with Java, updated edition, T. Budd, pearson education, 2000.
3. An Introduction to programming and OO design using Java, J.Nino and F.A. Hosch, John wiley & sons, 2001.

THEORY OF AUTOMATA & FORMAL LANGUAGE
(CS503)

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**
I	**Introduction:** 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.
II	**Regular Expression:** 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.
III	**Context free grammar (CFG) and Context Free Languages (CFL):** 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.
IV	**Push Down Automata (PDA):** 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.
V	**Turing machines (TM):** 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.

SOFTWARE ENGINEERING
(CS504)

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**
I	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.
II	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.
III	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: Halestead's Software Science, Function Point (FP) Based Measures, Cyclomatic Complexity Measures: Control Flow Graphs.
IV	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.
V	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.

DESIGN & ANALYSIS OF ALGORITHM LAB
(CS551)

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.

JAVA PROGRAMMING LAB
(CS552)

LIST OF EXPERIMENTS

1. Write a program to display the default value of all primitive data types in Java.
2. Write a Java program to sort given list of numbers.
3. Write a Java program to implement linear search.
4. Write a Java program to implement binary search.
5. Write a java program to add two given matrices.
6. Write a java program to multiply two given matrices.
7. Write a java program for sorting a given list of names.
8. Write a java program that checks whether a given string is a palindrome or not.
9. Write a java program that performs call by value and call by reference.
10. Write a java program that illustrates the simple inheritance.
11. Write a java program that illustrates the multilevel inheritance.
12. Write a java program that demonstrates the difference between method overloading and overriding.
13. Write a java program that demonstrates the difference between method overloading and constructor overloading.
14. Write a java program that describes the exception handling mechanism.
15. Write a java program that uses try & catch blocks and check whether the given array size is negative or not.
16. Write a java program that describes the user defined exception.
17. Write a java program that illustrates the creation of threads by using runnable class.
18. Write a java program that illustrates the multiple inheritances by using interfaces.
19. Write a java program to create a package named p1, and implement this package in ex1 class.
20. Write a java program to create a package named my pack, and import it in circle class.
21. Write a java program that illustrates the example for abstract class.
22. Write a java program that describes the life cycle of an applet. - A java program to Create a dialog box and menu. - A java program to create a grid layout control.
23. A java program to create a border layout control.
24. Write an Applet that creates a simple calculator.

SOFTWARE ENGINEERING LAB
(CS553)

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).

COMPUTER NETWORKS
(CS601)

Objective: The objective of this course is to provide basic exposure to computer networks theory and implementations.	
Unit	**Topic**
I	**Introduction:** Networks, Internet, Network Components, Network Categories, Applications of Computer Networks **Reference Models:** Concept of Layering, OSI Model, TCP/IP Protocol Suite, Functions of Layers **Physical Layer:** Transmission Mode, Physical Topology, Multiplexing, Transmission Media, Switching
II	**Data Link Layer:** Design Issues, Error Detection and Correction Techniques, Elementary Data Link Protocols, Sliding Window Protocols, Multiple Access Protocols, Ethernet, Connecting Devices
III	**Network Layer:** 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
IV	**Transport Layer:** 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.
V	**Application Layer:** 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

References:

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

COMILER DESIGN
(CS602)

Objective: Students will have a fair understanding of some standard passes in a general-purpose compiler.	
Unit	**Topic**
I	**Introduction to Compiler**: 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.
II	**Basic Parsing Techniques**: 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.
III	**Syntax-directed Translation:** 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.
IV	**Symbol Tables**: Data structure for symbols tables, representing scope information. **Run-TimeAdministration:** Implementation of simple stack allocation scheme, storage allocation in block structured language. Error Detection & Recovery: Lexical Phase errors, syntactic phase errors semantic errors.
V	**Code Generation**: 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, O‟Reilly Media, 1992.

ARTIFICIAL INTELLIGENCE
(CS603)

Objective: To learn the concepts of Artificial Intelligence and the methods of solving problems using Artificial Intelligence.	
Unit	**Topic**
I	**Introduction:** Introduction to Artificial Intelligence, Foundations and History of Artificial Intelligence , Application of Artificial Intelligence Communication - Communication among agents, natural language processing, formal grammar, parsing, grammar
II	**Introduction to Search**: Searching for solutions, Uniformed search strategies, Informed search Strategies, Local search algorithms and optimistic problems, Adversarial Search, Search for games, Alpha - Beta pruning.
III	**Knowledge Representation & Reasoning:** Propositional logic, Theory of first order logic, Inference in First order logic, Forward & Backward chaining, Resolution, Probabilistic reasoning, Utility theory, Hidden Markov Models (HMM), Bayesian Networks.
IV	**Decision making-** Utility theory, utility functions, and Decision theoretic Expert systems. Default reasoning, Fuzzy sets and fuzzy logic; AI languages and tools - Lisp, Prolog,
V	**Pattern Recognition :** Introduction, Design principles of pattern recognition system, Statistical Pattern recognition, Parameter estimation methods - Principle Component Analysis (PCA) and Linear Discriminant Analysis (LDA),

References:

1. Kevin Night and Elaine Rich, Nair B., “Artificial Intelligence (SIE)”, McGraw Hill, 2008.
2. Dan W. Patterson, “Introduction to AI and ES”, Pearson Education, 2007.
3. Peter Jackson, “Introduction to Expert Systems”, Pearson Education, 2011.

Python Programming
(CS604)

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

Unit	Topic
I	**Introduction To Python:** Installation and Working with Python Understanding Python variables Python basic Operators Understanding python blocks **Values and Variables :** Integer and String Values, Identifiers, User Input, String Formatting, Expressions and Arithmetic Examples
II	**Python Data Types:** 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
III	**Python Conditional Statements and looping:** If, If- else, Nested if-else For, While Nested loops
IV	**Python String, List And Dictionary Manipulations:** 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
V	**Python Object Oriented Programming:** 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

COMPUTER NETWORKS LAB
(CS651)

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
(CS652)

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[$a^nb/n>0$] . Test whether the following string belongs to this grammer.
10. Write a YACC & LEX program to identify valid if and if-else statement.

Python Programming LAB
(CS653)

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