Indian Institute of Technology Kanpur

COURSES OF STUDY

2025

Indian Institute of Technology Kanpur
KANPUR-208016

COMPUTER SCIENCE ENGINEERING

Template for the BT program in Computer Science and Engineering

Semester 1	Semester 2	Semester 3	Semester 4	Semester 5	Semester 6	Semester 7	Semester 8
SCHEME-1 (9) ELC111/ELC112/ ELC113 *	ETH111 (3) **	SCHEME-2 HSS-I (9-11)	SCHEME-3 EME (9-11)	SCHEME-4 HSS-II (9)	SCHEME-5 HSS-II (9)	SCHEME-6 HSS-II (9)	DE-8 (9)
MTH 111 (6)	MTH 113 (6)	ESC201 (14)	CS220 (13)	CS330 (13)	DE-2 (9-13)	DE-5 (9)	OE-4 (9)
MTH 112 (6)	MTH 114 (6)	ESO207 (12)	CS253 (12)	CS340 (9)	DE-3 (9)	DE-6 (9)	OE-5 (9)
PHY 114 (11)	PHY 113 (11)	CS201 (11)	CS202M (5)	CS345 (9)	DE-4 (9)	DE-7 (9)	OE-6 (9)
PHY 111 (3)	CHM 111 (3)	ESO-2 (9-11)	CS203M (5)	DE-1 (9)	OE-1 (9)	OE-3 (9)	OE-7 (9)
TA 111 (9)	ESC 111 (7)				OE-2 (5-9)		
CHM 112 (4)	ESC 112 (7)						
CHM 113 (4)	LIF111 (6)						
PE111 (3)	PE112 (3)						
55	52	54-58	44-46	49	50-58	45	45

Remarks

- ESO/SO courses are available in 6-14 credits each. ESO207A is compulsory for CSE students in the 3rd semester
- Students can take up to 4 UGPs, but only at most 3 UGPs (27 credits) can be counted towards graduation requirements.
- At least 2 DEs must be selected from Basket – A.
- All the four UGPs are optional.

Credit Table for BT Program in Computer Science and Engineering		
Course type	Recommended Credit range	Minimum Credits required for graduation
Institute Core (IC)	112	112
E/SO	18-45	21-23
Department requirements	144-179	149 (77 DC + 72 DE)
Open electives (OE)	51-57	59-63*
SCHEME	54-58	54-58
Total for 4-year BT/BS	391-420	395-405

* Exceeds the credit range recommended by UGARC

Template for the BTH program in Computer Science and Engineering

Template for 3rd to 8th semester for BTH program in Computer Science and Engineering					
Semester 3	Semester 4	Semester 5	Semester 6	Semester 7	Semester 8
SCHEME-2 HSS-I (9-11)	SCHEME-3 EME (9-11)	SCHEME-4 HSS-II (9)	SCHEME-5 HSS-II (9)	SCHEME-6 HSS-II (9)	DE-8 (9)
ESC201 (14)	CS220 (13)	CS330 (13)	DE-2 (9-13)	DE-5 (9)	OE-4 (9)
ESO207 (12)	CS253 (12)	CS340 (9)	DE-3 (9)	DE-6 (9)	OE-5 (9)
CS201 (11)	CS202M (5)	CS345 (9)	DE-4 (9)	DE-7 (9)	OE-6 (9)
ESO-2 (9-11)	CS203M (5)	DE-1 (9)	OE-1 (9)	OE-3 (9)	OE-7 (9)
	DEH-1 (9)	DEH-2 (9)	OE-2 (5-9)	DEH-3 (9)	
54-58	53-55	58	50-58	45	45

- CPI Criteria for BTH: Above 8.5

Remarks

- ESO/SO courses are available in 6-14 credits each. ESO207 is compulsory for CSE students in the 3rd semester.
- At least 2 UGPs are compulsory (DE) for BTH
- At least 27 credits of DEH should be completed by taking CS6XX and CS7XX level courses
- Students can take up to 4 UGPs, but only at most 3 UGPs (27 credits) can be counted towards graduation requirements.
- At least 2 DEs must be selected from Basket – A.

Template for the BTM program in Computer Science and Engineering

Template for 3rd to 8th semester for BTM program in Computer Science and Engineering					
Semester 3	Semester 4	Semester 5	Semester 6	Semester 7	Semester 8
SCHEME-2 HSS-I (9-11)	SCHEME-3 EME (9-11)	SCHEME-4 HSS-II (9)	SCHEME-5 HSS-II (9)	SCHEME-6 HSS-II (9)	OE-2 (9)
ESC201 (14)	CS220 (13)	CS330 (13)	DE-2 (9-13)	DE-4 (9)	OE-3 (9)
ESO207 (12)	CS253 (12)	CS340 (9)	DE-3 (9)	DE-5 (9)	OE-4 (9)
CS201 (11)	CS202M (5)	CS345 (9)	MTB-1 (9)	MTB-3 (9)	MTB-5 (9)
ESO-2 (9-11)	CS203M (5)	DE-1 (9)	MTB-2 (9)	MTB-4 (9)	MTB-6 (9)
			OE-1 (5-9)		
54-58	44-46	49	50-58	45	45

Remarks

- ESO/SO courses are available in 6-14 credits each. ESO207A is compulsory for CSE students in the 3rd semester.
- Students can take up to 4 UGPs, but only at most 3 UGPs (27 credits) can be counted towards graduation requirements.
- At least 2 DEs must be selected from Basket – A.
- All the four UGPs are optional.

Template for five-year dual degree program in Computer Science and Engineering

Template for semester 3 to 6 same as the BT program

Semester 7	Semester 8	Summer	Semester 7	Semester 7
SCHEME-6, HSS-II (9)	DE PG-1 (9)	M. Tech Thesis (9)	DE PG-4 (9)	DE PG-6 (9)
DE-5 (9)	DE PG-2 (9)		DE PG-5 (9)	M. Tech Thesis (36)
OE-2 (9)	DE PG-3 (9)		M. Tech Thesis (27)	
DE-6 (9)	M. Tech Thesis (9)			
DE-7 (9)				
DE-8 (9)				
54	36	9	45	45

MINIMUM CREDIT REQUIREMENT IN M. TECH FOR GRADUATION

PG Component: 54 credits Thesis

Component: 81 credits

REMARKS:

- Up to 36 OE credits may be used from the BT minimum requirements to fulfil requirements for the BT-MT dual degree programme. These will be waived from the BT programme and counted towards PG requirements.
- All other minimum BT credit requirements need to be fulfilled, including those that are slotted in the 7th and 8th semester of the BT template.

Template for double major: second major in Computer Science and Engineering

Odd Semester	Even Semester
CS330 (13)	CS202M (5) and CS203M (5)
CS340 (9)	CS220 (13)
CS345 (9)	CS253 (12)
CS DE-1 (9)	CS DE-2 (9-13)
CS DE-3 (9)	CS DE-4 (9)
49	53-57

- Total mandatory credits for second major in computer science: 102

Course and credit Waiver policy: Following waivers will be granted automatically:

– Course and credits to be waived If the student has done

CS340	MTH401
CS202	MTH302
CS203	MSO201 OR HSO201

- To clarify, if a student gets waiver for all the three courses above, he will need to do only 102- 19=83 credits to finish his/her CSE second major requirement. If a student gets a waiver for CS202 only, he will need to do 102-5=97 credits to finish his/her CSE second major requirement. The table above may be extended in future if other courses similar to those in CSE major template are added/discovered in the other departments. Any other course waivers (without credit waivers) may be recommended by CSE DUGC based on its discretion.

REMARKS:

- Two DEs should be selected from Basket-A (Details of Basket-A are available in CSE B.Tech. template).
- Total CSE-DE credits should be at least 36.
- Up to 36 OE credits may be waived from the parent department BT/BS graduation requirements when they are used to fulfil requirements for the double major.
- *ESO207 is a compulsory course to be done, but its credits will be counted against the ESO requirement of the first major.

Minors in Computer Science and Engineering The

minors offered by CSE are listed in the table below.

- For each minor stream, the minimum total credit requirement is 30
- In each minor stream, any related course(s), if not mentioned in the list of optional courses, may be taken as optional with the permission of the CSE DUGC

The course ESO207 can be waived if the student has done an equivalent course. CSE DUGC convener can give more information about it.

DEPARTMENT OF CSE

Courses ID	Course Title	Credits L-T-P-D-[C]	Content
CS201	Mathematics For Computer Science - I	3-0-0-0-4	Sets, Proofs: [Weeks 1] Sets, relations, functions, countable and uncountable sets. Proofs; Proofs by deduction, contrapositive and contradiction (diagonalization). Proofs by induction * All these proof techniques can be covered through examples. The rigorous notion of proof will be covered in CS 202 and can be skipped here. Basic Counting: [Weeks 2] Selection/combination, arrangements/permutation, rule of sum, rule of product. Binomial coefficients, identities, multinomial coefficients. Selection with repetition/distributions of objects into cells, distinguishable/indistinguishable objects. Combinatorial problems with restrictions Generating Functions: [Week 3] Recurrence relations to solve combinatorial problems. Generating functions to solve recurrence. Counting Techniques: [Weeks 4] Inclusion-exclusion, Pigeonhole principle, Ramsey's theorem Partial Order: [Week 5] Equivalence relations, partitions, partial order, posets, chain/antichain. Graph Theory: [Week 6 And 7] Definitions, degree, paths, cycles, Hamiltonian path, Eulerian cycles. cycles and acyclic graphs. Trees, spanning trees, networks. Number Theory: [Week 8 And 9] Divisibility, primes, division theorem, Euclid's gcd/extended Euclid's algorithm, Unique factorization domain. Modular arithmetic, sums and products, Chinese remaindering, Mobius inversion. RSA: [Week 10] Fermat's little theorem, Euler's theorem. Application: RSA. Finite Fields: [Week 11] Z_p, the cyclic structure of Z_p^*. Definition of the field as a generalization to F_p. Application: Polynomials over F_p, error correction. Group Theory: [Week 12 And 13] Definitions, examples (Z_n and Z_n^*) cyclic and dihedral group, abelian groups.

			Subgroups, cosets, partition permutation group, transpositions, cycle representation symmetries as a group. Optional Topics: [If Time Permits] Rings Applications Of Group Theory (Burnside lemma and generalization to Polya's theorem, use of group theory in combinatorics). Other interesting applications.
CS202M	Mathematics For Computer Science - II	3-0-0-9	Propositional logic syntax and semantics. Tautologies, axiom system and deduction. Proof of soundness and completeness. First order logic syntax and semantics. Structures, models, satisfaction and validity. Axiomatization, soundness and completeness. Optional: some advanced topics.
CS203M	Mathematics for Computer Science - III	3-0-0-9	Basics Of Probability Theory : [Weeks 1] Sets, The concept of a discrete sample space in probability theory. The definition of an event. The definition of a probability distribution. De Morgan’s Law, Union Bounds Distributions : [Weeks 2] Random variables, expectation, and variance. Discrete distributions: Bernoulli trials, Geometric, Binomial and Hypergeometric and Negative Binomial distributions, Poisson distribution. Continuous distributions: normal and other continuous distributions. Exercises based on the analysis of applications to computer science. Linearity of expectation. Higher moments of a random variable, moment generating function. Computing the moments of geometric, binomial, normal, and Poisson distributions. OPTIONAL (if time permits): Function of One Random Variable: Change of Variables. Conditional Probability : [Weeks 3] Conditional Probability, Conditional expectation of a random variable with respect to an event. Bayes' Theorem and examples of applications in computer science. Independence: [Weeks 4] The concept of k-wise and mutual independence of random variables. Applications of independence and k-wise independence in computer science. Tail Bounds: [Week 5]

			Tail bounds: Markov inequality, Chebyshev's Inequality, Chernoff bound, and Examples of applications to the analysis of randomized algorithms. Applications: [Week 6 & 7] Cover one or more applications: For example, Statistics: Hypothesis Testing Parameter Estimation: MLE, Least Squares Introduction to the probabilistic method. Applications to random graphs and number theory. Lovasz Local Lemma and applications Information theory Markov chains Randomized algorithms/ Streaming
CS220	Computer Organisation	3-0-3-12	Introduction. Arithmetic algorithms. Overview of basic digital building blocks; truth tables; basic structure of a digital computer. Number representation: Integer - unsigned, signed (sign magnitude, 1â€™s complement, 2â€™s complement); Characters - ASCII coding, other coding schemes; Real numbers - fixed and floating point, IEEE754 representation. Basic building blocks for the ALU: Adder, Subtracter, Shifter, Multiplication and division circuits. Hardware description language. Introduction to some HDL (Verilog, VHDL, BSV). Digital Design using HDLs. CPU. CPU Sub-block: Datapath - ALU, Registers, Instructions, Execution of Instructions; CPU buses; Control path - microprogramming, hardwired logic; External interface. Advanced Concepts: Pipelining; Introduction to Advanced Processors (multiprocessors and multi-cores). Examples of some well known processors. Assembly Language Programming. Instruction set and Assembly programming for some processor, preferably the one described in class. Memory. Memory Sub-block: Memory organization; Technologies - ROM, RAM, EPROM, Flash, etc., Virtual Memories. Cache: Cache algorithms, Cache Hierarchy, Cache coherence protocols. Advanced concepts: Performance, Interleaving, On chip vs Off chip Memories/Caches. I/O and Peripherals. I/O Sub-block: I/O techniques - interrupts, polling, DMA; Synchronous vs. Asynchronous I/O; Controllers.

			Peripherals: Keyboard, Mouse, Monitors, Disk drives, etc. Lab Contents. Digital Design using HDLs. Simple circuit designs: For e.g. Counter, Multiplexer, Arithmetic circuits etc. Design of a Simple Processor: Includes register file, ALU, data paths. FPGA Programming Programming on Xilinx Spartan 3E (or equivalent) FPGA. Handling of Inputs: through slide switches, through push buttons. Handling of Outputs: 7-segment display, LED display, LCD display. The designs developed in Part-I can be used to program the FPGA. Assembly Language Programming Programming in assembly language. The assignments should cover the following concepts: Registers; different type of instructions (load, store, arithmetic, logic, branch); operand addressing modes; memory addressing modes; conditions (codes/flags and conditional branches) stack manipulation; procedure calls; procedure call conventions (load/store of; arguments on stack, activation records);
CS251	Computing Laboratory - I	0-0-3-3	Basic operating system commands. Students are expected to know the basic shell (e.g., bash) commands and should be able to understand the options and functioning of a command by reading the man and info pages. Editors. Again, students are expected to be familiar with at least one of the two editors vim and emacs. However, they should be able to utilize the multiple features of the editors (such as automatic indentation, context-sensitive colouring, letype-sensitive auto-wrap, etc.) and not use them simply as a typewriter. Version control. Students will need to completely know how at least one of the version control systems (e.g., cvs, svn, git, darcs) work. They should be able to check in, check out, resolve errors and put tags on a snapshot. On all subsequent assignments, they must use a form of version control. Scripting and automation. Of the various types of shells (bash, csh, tcsh, ksh), the preferred choice is bash, although students should be familiar with the different command syntax in other shells as well. Also, they will need to know the various functions (e.g., seq, for) that a shell provides. The choice of the scripting language (perl or python) is open to students. Document preparation. Students will learn using latex for preparing documents. They should also know how to format properly the equations, gures, tables, theorems, etc. using different packages and options. For bibliography management, they should use bibtex, and must use it within the latex

			documents. For drawing figures and graphs, they can choose to learn some or all of the different softwares used popularly (they include gnuplot, xfig, etc.). Hardware. Students will work hands-on to learn how to install hard drives, RAM, etc., and in general, assemble a computer from its different parts. Web applications. Students will need to know the different web languages (html, xml) that are used along with the scripting languages (php, javascript), forms and database tools (MySQL) that are necessary for setting up a website. Students should be able to setup a wiki site as well. Useful Application Software. Students should learn to effectively use various softwares that serve a whole range of applications. These include the general purpose octave or scilab, the more statistically oriented R and the image manipulator gimp. Operating system installation and packages Students can choose a particular distribution of Linux (e.g., Fedora, Debian, Ubuntu, etc.) and learn installing the basic operating system and different packages. They should also learn how to congure options globally (including booting) and for particular users only. They will further understand how to debug problems using the log and error les.
CS252	Computing Laboratory - II	0-0-3-3	System administration: Students will learn to setup and manage a network server including a web server and an email server. They should also be familiar with various network protocols. Further, students will be able to administer different components of the system by using various monitoring tools. They should also learn simple load-balancing tools. Security: While setting up different network and database servers, students will learn to manage the security issues including different attacks. Public key infrastructure (PKI) is a good example of how to setup, manage, and distribute certicates that are issued as authorization tools. As part of maintaining sys- tems, students will need to use system vulnerability and intrusion testing tools as well. Compiler tools: Students will need to learn the various low-level tools used routinely in compilers including lex and yacc. Programming environment tools: For efcient programming, students are expected to use various integrated development environments (IDEs) such as eclipse and debuggers such as gdb. They should also use tools for effective tagging and browsing of source code. Finally, they will learn to use the build tools that are necessary for large software projects.

CS253	Software Development And Operations	**3-0-3-12**	Instructor: Dr. Sandeep K. Shukla / Dr. T. V. Prabhakar /Dr. Indranil Saha/Dr. Biswabandan Panda/Dr. Debadatta Mishra/Dr. Preeti Malakar/Dr. Amey Karkare Major, Measurable Learning Objectives Having successfully completed this course, the student will be able to: Learn how to develop large scale software from scratch Develop a full-fledged software development environment with knowledge of utilities such as shell scripting make file creation, regular expression tools, and other development tools such as autoconfiguration tools Develop testing harnesses and testing techniques including unit testing, integration testing, and system level testing Develop deployment environments, and software maintenance tools, and updating tools Practice defensive and secure programming with awareness of various vulnerabilities at the code level, software design level, architecture level Use software libraries in large scale software development - networking libraries, data base connectivity libraries, and GUI libraries etc. Develop software design documents and test documents Prerequisites And Co-Requisites: ESC 101, ESO 207 Texts And Special Teaching Aids Course notes, lecture slides, man pages, and other documentations will be provided to students by the instructor. Self-study would be an important component of this course. Syllabus The students will be exposed to the following topics: Regular Expression Tools (e.g Awk) Lex and Yacc Bash Scripting Basic Perl, Python or Ruby Scripting Make File Details and Autoconfiguration tools, gdb Object Oriented Programming (C++ or Java) Unit Testing Tools and Techniques Integration and System Level Testing Techniques Software Requirements Analysis, Requirements Generation, Documentation Software Architecture, Functional, Performance and Security Specification Software Design with Object Oriented Design Techniques, Design Documentation Implementation techniques with defensive programming and secure programming techniques Unit testing

			Development and Integration and System Testing, Test Automation Deployment Issues and Maintenance issues Software Quality Metrics and measurements The course will consist of 3 hours of lectures per week, projects and homework, and a course project.
CS300	Technical Communication	0-0-2-2	Technical writing Editing your own work Critiquing others' work, mentoring Preparing and making presentations
CS315	Principles Of Database Systems	3-0-0-9	Introduction: Database applications, purpose, accessing and modifying databases, need for transactions, architecture - users and administrators, data mining, information retrieval. iRelational Databases: relational model, database schema, keys, relational query languages, algebra, tuple and domain calculus example queries, (optional: equivalence of relational calculus and relational algebra). SQL: Data definition, basic SQL query structure, set operations, nested subqueries, aggregation, null values, database modification, join expressions, views. Database Design: E-R model, E-R diagram, reduction to relational schema, E-R design issues, database integrity, specifying integrity constraints in SQL: unique columns, foreign key, triggers. Relational Database Design: features of good design, Functional Dependency theory, decomposition using functional dependency and normal forms, algorithms for decomposition, normal forms, (optional: multi-valued dependency and 4th normal form). Storage and File structure: Overview of secondary storage, RAID and flash storage. Storing tables: row-wise, column database, database buffer. Indexing: concepts, clustered and non-clustered indices, B+-tree indices, multiple key access, hashed files, linear hash files, bitmap indices, Index definition in SQL, ++R-trees. Query Processing: Overview, measures of query cost, selection, sorting, join processing algorithms-nested loops, merge-sort, hash join, aggregation. Query Optimization: purpose, transformation of relational expressions, estimating cost and statistics of expression, choosing evaluation plans, linear and bushy plans, dynamic programming algorithms. Transactions: Concept and purpose, ACID properties and their necessity, transactions in SQL. Problems with full isolation and levels of isolation. Concurrency Control: lock-based protocols, 2-phase locking, deadlock handling, multiple granularity, timestamp based protocols, index locking, (optional: validation protocols, multi-version protocols, snap

			shot isolation, predicate locking, concurrency control for index structures). Recovery: Failures and their classification, recovery and atomicity, recovery algorithms, Undo-Redo with write ahead logging, no Undo no Redo and other combinations, buffer management, (optional: ARIES recovery). **Optional/Advanced topics below covered at the discretion of instructor** Parallel Databases: Avenues for parallelism: I/O parallelism, interquery, inter-query and intra operation parallelism, databases for multi-core machines. Distributed Databases: Distributed data storage, distributed transactions, commit protocols, concurrency control in distributed databases, heterogeneous and cloud-based databases. Data Mining: Decision Support Systems, data warehousing, mining, classification, association rules, clustering Information Retrieval: relevance ranking using terms and hyperlinks,page rank, indexing of documents, measuring retrieval effectiveness. XML and semi-structured data: necessity, XML document schema, querying: XPath and XQuery languages, applications.
CS330	Operating Systems	3-0-3-12	Introduction: review of computer organization, intoduction to popular operating systems like UNIX, Windows, etc., OS structure, system calls, functions of OS, evolution of OSs. Computer organization interface: using interrupt handler to pass control between a running program and OS. Concept of a process: states, operations with examples from UNIX (fork, exec) and/or Windows. Process scheduling, interprocess communication (shared memory and message passing), UNIX signals. Threads: multithreaded model, scheduler activations, examples of threaded programs. Scheduling: multi-programming and time sharing, scheduling algorithms, multiprocessor scheduling, thread scheduling (examples using POSIX threads). Process synchronization: critical sections, classical two process and n-process solutions, hardware primitives for synchronization, semaphores, monitors, classical problems in synchronization (producer-consumer, readers-writer, dining philosophers, etc.). Deadlocks: modeling, chararcterization, prevention and avoidance, detection and recovery. Memory management: with and without swapping, paging and segmentation, demand paging, virtual memory, page replacement algorithms, working set

			model, 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. operations on them, UNIX FS, UFS protection and security, NFS. Protection and security: Illustrations of security model of UNIX and other OSs. Examples of attacks. Epilogue: Pointers to advanced topics (distributed OS, multimedia OS, embedded OS, real-time OS, OS for multiprocessor machines). All the above topics will be illustrated using UNIX and/or Windows as case-studies. The lectures will be supplemented by a set of assignments/projects on an instructional operating system. This is the lab component.
CS335	Compiler Design	3-0-3-12	Compiler structure: analysis-synthesis model of compilation, various phases of a compiler, tool based approach to compiler construction. Lexical analysis: interface with input, parser and symbol table, token, lexeme and patterns. Difficulties in lexical analysis. Error reporting. Implementation. Regular definition, Transition diagrams. Syntax analysis: CFGs, ambiguity, associativity, precedence, top down parsing, recursive descent parsing, transformation on the grammars, predictive parsing, bottom up parsing, LR parsers (SLR, LALR, LR). Syntax directed definitions: inherited and synthesized attributes, dependency graph, evaluation order, bottom up and top down evaluation of attributes, L- and S-attributed definitions. Type checking: type system, type expressions, structural and name equivalence of types, type conversion, overloaded functions and operators, polymorphic functions. Run time system: storage organization, activation tree, activation record, stack allocation of activation records, parameter passing mechanisms. Intermediate code generation: intermediate representations, translation of declarations, assignments, control flow, boolean expressions and procedure calls. Implementation issues. Code generation and instruction selection: issues, basic blocks and flow graphs, register allocation, code generation, dag representation of programs, code generation from dags, peep hole optimization, code generator generators, specifications of machine.

			Introduction to Dataflow Anaysis (Reaching Definitions and Live Variable Analysis). Introduction to compilation for modern architectures (superscaler out-of-order, VLIW, GPU etc.).
CS340	Theory Of Computation	3-0-0-9	Introduction: Motivation for studying theory of computation, a quick overview of the subject. Notion of formal language. Language membership problem, why this is taken as the central problem of the subject. Finite automata and regular expressions: DFA, NFA (with and without transitions), their equivalence. Proof that for some languages NFAs can be exponentially more succinct than DFAs. Denition of regular expressions. Proof that FAs recognize, and regular expressions denote the same class of languages, viz., regular languages. Properties of regular languages: Pumping lemma and its use to prove non-regularity of a language, closure properties of class of regular languages, decision properties: convert- ing among representations, testing emptiness, etc. Minimization of DFAs, Myhill-Nerode theorem. Context-free grammars and languages: Derivation, parse trees. Language generated by a CFG. Eliminating useless symbols, -productions, unit productions. Chomsky normal form. Pushdown automata: Denition, instantaneous description as a snapshot of PDA com- putation, notion of acceptance for PDAs: acceptance by nal states, and by empty stack; the equivalence of the two notions. Proof that CFGs generate the same class of languages that PDAs accept. Properties of context-free languages: Pumping lemma for context-free languages and its use to prove a language to be not context-free. Closure properties of the class of context- free languages. CYK algorithm for CFL membership, testing emptiness of CFLs. Turing machines: Historical context, informal proofs of undecidability. Denition of TM, instantaneous description as a snapshot of TM computation, notion of acceptance. Robustness of the model: both natural generalizations and restrictions keep the class of languages accepted invariant. (Generalizations: multi-track, multi-tape, nondeterministic, etc. Restrictions: semi-innite tape, counter machines). Church-Turing hypothesis. Undecidability: Denitions of r.e. and recursive languages. Turing machine codes, the diagonalization language and proof that it is not r.e. Universal Turing machine. Universal language, its semi-decidability. Reducibility and its use in proving undecidability. Rices theorem. Undecidability of Posts correspondence problem.

			Intractability: Motivation for the notion. The class P as consensus class of tractable sets. Classes NP, co-NP. Polynomial time reductions. NP-completess, NP-hardness. Cook- Levin theorem. Mention about boundary of tractability: 2SAT vs. 3SAT, 2D matching vs. 3D matching. Some NP-completeness proofs: vertex cover, clique, independent sets, Hamiltonian graphs, subset-sum, set cover.
CS345	Algorithms II	3-0-0-9	Amortized analysis. Exposure to some advanced data structures (For example, Fibonacci heaps or augmented data structures or interval trees or dynamic trees). As part of CS210/ESO211 course, three algorithm paradigms, namely, greedy method, divide and conquer, and dynamic programming are discussed. These algorithm paradigms should be revisited in CS345, but with advanced applications and/or emphasis on their theoretical foundations as follows. Greedy method: theoretical foundations of greedy method (matroids) and other applications. Divide and Conquer: FFT algorithm and other applications. Dynamic Programming: Bellman Ford algorithm and other applications. Graph algorithms: all-pairs shortest paths, biconnected components in undirected graphs, strongly connected components in directed graphs, and other problems. Pattern matching algorithms. Lower bound on sorting. Algorithms for maximum flow and applications. Notion of intractability: NP-completeness, reduction (the proof of Cook-Levin theorem may be skipped) Exposure to some (one or more) topics from the following list : Approximation algorithms. Algebraic and number theoretic algorithms. Computational Geometry. Linear programming. Parallel/distributed algorithms. Randomized algorithms.
CS350	Principles Of Programming Languages	3-0-0-9	Imperative Languages: block structure, scope rules, parameter passing, constructs like coroutines, tasks, exceptions etc. Functional programming: functions, recursion, macros, user-defined control constructs, higher order constructs, types, data abstraction, lazy evaluation, polymorphism, semantics, type inference, and implementation issues. Two of the following three topics may be covered in detail. Declarative programming: declarative programming, Horn clauses, procedural interpretation of Horn clauses, SLD-resolution including unification, the

			logical variable, implementation issues: abstract machines and compiling to abstract machines. Declarative Concurrency: Data-driven concurrent model, basic thread programming techniques, Streams, lazy execution, Message passing concurrency models. Object-oriented programming: objects and programming with objects, classes and instances, hierarchies and inheritance, encapsulation, semantics of OO languages and implementation issues.
CS360	Introduction To Computer Graphics And Simulation		Introduction to Picture Synthesis and Analysis. Conceptual Framework of an Interactive Graphical Simulation System. Graphics hardware. Basic Raster Graphics Algorithms. Introduction to Simple Raster Graphics Package (SRGP). Graphics Entities. Geometric Transformations. Object hierarchy. Segmentation. Interaction Techniques. Geometric Modeling in 3-D. Viewing in 3-D. Concept of Synthetic Camera. Dialogue Design. Graphics User Interfaces. Windowing Systems. Graphical Modeling of Discrete events. Simulation of Discrete Event Displays. Animation Techniques. Basic Rules for Animation. Graphical Simulation of continuous motion. Role of Virtual Reality in Graphical Simulation.
CS365	Artificial Intelligence	3-0-0-9	AI: Introduction Brief history. Agents and rationality, task environments, agent architecture types. Search and Knowledge representation. Search spaces. Uninformed and informed search. Hill climbing, simulated annealing, genetic algorithms. Logic based representations (PL, FoL) and inference, Prolog. Rule based representations, forward and backward chaining, matching algorithms. Probabilistic reasoning and uncertainty. Bayes nets and reasoning with them. Uncertainty and methods to handle it. Learning. Forms of learning. Statistical methods: naive-Bayes, nearest neighbour, kernel, neural network models, noise and overfitting. Decision trees, inductive learning. Clustering - basic agglomerative, divisive algorithms based on similarity/dissimilarity measures. Applications to NLP, vision, robotics, etc.

<table>
<tr><td>CS395</td><td>Undergraduate Project - I (UGP-1)</td><td></td><td></td></tr>
<tr><td>CS396</td><td>Undergraduate Project - II (UGP-2)</td><td></td><td></td></tr>
<tr><td>CS397</td><td>Special Topics In Computer Science</td><td></td><td>This course is meant for a 3rd year BTech (CSE) student to study a topic of their interest, somewhat independently. A student may also carry out a project in this course.

In this course, there will be a faculty member associated with each student whose responsibility will be to suggest reading material, hold discussion sessions, monitor the progress of the student, examine the student, and give a grade at the end of the semester.</td></tr>
<tr><td>CS422</td><td>Computer Architecture</td><td>3-0-0-9</td><td>Overview of computer architecture, performance evaluation of processors.
Pipelining, super-pipelines, advanced pipelines, static and dynamic scheduling, instruction-level parallelism, loop unrolling, VLIW and superscalar processors, vector processing and array processing.
Memory bandwith issues, memory organization, cache hierarchy.
Symmetric multiprocessors (SMP), NUMA-MPs, massively parallel processors, cache coherence protocols, interconnection networks, I/O processing, multiprocessing, multiplexing, examples of contemporary architectures, RAS (Reliability, Availability, Scalability) features.</td></tr>
<tr><td>CS423</td><td>Multi-Core And Multiprocessor Architecture</td><td>3-0-0-0 (9)</td><td>Objectives:
The primary objective of the course is to discuss the principles and practices of the design of the contemporary multi-core and multiprocessor architectures.

Summary:
This course studies the principles and practices of multi-core and multiprocessor design. It introduces students to the broad topics such as cache coherence, memory consistency models, synchronization primitives, on-chip interconnection networks, and performance pathologies of shared memory parallel programs.

Contents:
<table>
<tr><th>Sl. No.</th><th>Broad Title</th><th>Topics</th><th>No. of Lectures (Each 75 minutes)</th></tr>
</table>
</td></tr>
</table>

			1.	Introduction	Multi-cores: why and what; Moore's law; Dennard scaling	2
			2.	Fundamentals of memory system	Virtual memory; address translation hardware; SRAM and caches; DRAM and main memory	6
			3.	Tools and techniques for evaluating architectures	Simulation; dynamic binary instrumentation; performance counters; use of special instructions such as cupid of x86	3
			4.	Introduction to shared memory multiprocessors and multi-cores	Types of architectures; problem of cache coherence; specification of cache coherence protocols as a set of invariants; basics of memory consistency models	5
			5.	Shared memory synchronization	Hardware support for efficient synchronization; interplay of cache coherence, speculative execution, and synchronization primitives; implementation of efficient locks and barriers	3
			6.	Performance analysis of shared memory parallel programs	Brief introduction to shared memory parallel programming techniques: POSIX thread model, OpenMP, fork/mmap; performance pathologies of shared memory parallel programs; influence of cache coherence and synchronization	3
			7.	Scalable cache coherence	Directory-based coherence protocols and their implementation; case study of SGI Origin 2000 protocol	3
			8.	Memory consistency models	Sequential consistency, total store order, partial store	2

<table>
<tr><td></td><td></td><td></td><td>
<table>
<tr><td></td><td></td><td>order, processor consistency, weak ordering, release consistency</td><td></td></tr>
<tr><td>9.</td><td>Interconnection networks</td><td>Topologies, integrated router design, routing techniques for networks on chip, interplay of deadlockfree routing and cache coherence; virtual channels</td><td>2</td></tr>
</table>
</td></tr>
<tr><td>CS425</td><td>Computer Networks</td><td>3-0-0-9</td><td>Introduction, history and development of computer networks, networks topologies. Layering and protocols.
Physical Layer: Different types of transmission media, errors in transmission: attenuation, noise. Repeaters. Encoding (NRZ, NRZI, Manchester, 4B/5B, etc.).
MAC Layer: Aloha, CSMA, CSMA/CD, CSMA/CA protocols. Examples: Ethernet, including Gigabit Ethernet and WiFi (802.11). Time permitting, a quick exposure to Token Ring and to Bluetooth, WiMax may also be included.
Data Link Layer: Error detection (Parity, CRC), Sliding Window, Stop and Wait protocols.
LAN: Design, specifications of popular technologies, switching. A student should be able to design LAN of a campus or a building.
Network layer: Internet Protocol, IPv6, ARP, DHCP, ICMP, Routing algorithms: Distance vector, Link state, Metrics, Inter-domain routing. Subnetting, Classless addressing, Network Address Translation.
Transport layer: UDP, TCP. Connection establishemnt and termination, sliding window revisited, flow and congestion control, timers, retransmission, TCP extensions, etc.
Design issues in protocols at different layers.
Network Programming: Socket Programming.
Session, Presentation, and Application Layers. Examples: DNS, SMTP, IMAP, HTTP, etc.
Network Security: Conepts 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.</td></tr>
<tr><td>CS433</td><td>Parallel Programming</td><td>3-0-0-9</td><td>Introduction: Why parallel computing; Ubiquity of parallel hardware/multi-cores; Processes and threads; Programming models: shared memory and message passing; Speedup and efciency; Amdahls Law.
Introduction to parallel hardware: Multi-cores and multiprocessors; shared memory and message</td></tr>
</table>

			passing architectures; cache hierarchy and coherence; sequential consistency. Introduction to parallel software: Steps involved in developing a parallel program; Depen- dence analysis; Domain decomposition; Task assignment: static and dynamic; Performance issues: 4C cache misses, inherent and artifactual communication, false sharing, computation-to-communication ratio as a guiding metric for decomposition, hot spots and staggered communication. Shared memory parallel programming: Synchronization: Locks and barriers; Hardware primitives for efcient lock implementation; Lock al- gorithms; Relaxed consistency models; High-level language memory models (such Java and/or C++); Memory fences. Developing parallel programs with UNIX fork model: IPC with shared memory and message pass- ing; UNIX semaphore and its all-or-none semantic. Example case studies (see note below for some details). Developing parallel programs with POSIX thread library: Thread creation; Thread join; Mutex; Condition variables. Example case studies (see note below for some details). Developing parallel programs with OpenMP directives: Parallel for; Parallel section; Static, dy- namic, guided, and runtime scheduling; Critical sections and atomic operations; Barriers; Reduction. Example case studies (see note below for some details). Message passing programming: Distributed memory model; Introduction to message passing interface (MPI); Synchronization as Send/Recv pair; Synchronous and asynchronous Send/Recv; Collective communication: Reduce, Broadcast, Data distribution, Scatter, Gather; MPI derived data types. Example case studies (see note below for some details). Introduction to GPU programming: GPU architecture; Introduction to CUDA programming; Concept of SIMD and SIMT computation; Thread blocks; Warps; Global memory; Shared memory; Thread divergence in control transfer; Example case studies (see note below for some details). Additional topics: PGAS and APGAS programming paradigms; Transactional memory paradigm; Introduction to speculative parallelization. ** Notes **: The example case studies should be chosen to cover a wide variety of parallel algorithms drawn from nu- meric as well as non-numeric domains. Possibilities include parallel sort, parallel prex, parallel search, graph algorithms, parallel ranking, reduction, algorithms using tree, fan, pipe

			paradigms, matrix computa- tion, equation solvers, n-body simulation, ray tracing, etc. The instruction must accompany an adequate number of programming assignments demonstrating the concepts. The instructors are encouraged to offer large semester-long programming projects.
CS455	Software Engineering	3-0-0-9	Software development lifecycle. Process models. Requirements specifications. Basic software architecture. Software design, UML modelling. Design patterns in software. Software implementation. Testing, verification and validation. Static analysis. Introduction to software model checking. Software metrics. Software project management.
CS497	Special Topics In Computer Science	3-0-0-0-4	This course is meant for a 4th year BTech (CSE) student to study a topic of their interest, somewhat independently. A student may also carry out a project in this course. In this course, there will be a faculty member associated with each student whose responsibility will be to suggest reading material, hold discussion sessions, monitor the progress of the student, examine the student, and give a grade at the end of the semester.
CS498	Undergraduate Project - III (UGP-3)		First semester project work.
CS499	Undergraduate Project - IV (UGP-4)		Second semester project work.
CS601	Mathematics For Computer Science		Linear Algebra: Fields. Vectors spaces, examples,Rn, Cn; subspaces. Linear independence, dependence and dimension. Linear transformations. Matrices, matrix algebra, determinants. Properties of matrices and determinants. Systems of linear equations. Eigenvalues, eigenvectors, eigenspaces, diagonalization and the spectral theorem. Factorization and singular value decomposition. Probability: Sample spaces, events, axioms of probability. Conditional probability and independence. Random variables. Discrete and continuous random variables, densities and distributions. Expectation and its properties.

			Normal distribution and its properties. Law of large numbers, central limit theorem. Bounds on deviations: Chebyshev, Markov, Hoeffding, Chernoff. Introduction to Markov chains, random walks. Logic: What is a proof? And proof methods. Propositional logic syntax and semantics. Tautologies, axiom system and deduction. Proof of soundness and completeness. First order logic syntax and semantics. Converting natural language into FoL wffs. Structures, models, satisfaction and validity. Axiomatization, soundness and completeness. Refutation and logic programming.
CS602	Design And Analysis Of Algorithms		Sorting: Review of various sorting algorithms, topological sorting. (2 lecture) Graph Definitions and Elementary Algorithms: Shortest path by BFS, shortest path in edge-weighted case (Dijkasra's), depth-first search and computation of strongly connected components, emphasis on correctness proof of the algorithm and time/space analysis, example of amortized analysis. (4 lectures) Matroids: Introduction to greedy paradigm, algorithm to compute a maximum weight maximal independent set. Application to MST. (3 lecture) Graph Matching: Algorithm to compute maximum matching. characterization of maximum matching by augmenting paths, Edmond's Blossom algorithm to compute augmenting path. (3 lectures) Flow-Networks: Maxflow-mincut theorem, Ford-Fulkerson Method to compute maximum flow, Edmond-Karp maximum-flow algorithm. (3 lectures) Matrix Computations: Strassen's algorithm and introduction to divide and conquer paradigm, inverse of a triangular matrix, relation between the time complexities of basic matrix operations, LUP-decomposition. (3 lectures) Shortest Path in Graphs: Floyd-Warshall algorithm and introduction to dynamic programming paradigm. More examples of dynamic programming. (3 lecture) String Matching: Knuth-Morris-Pratt algorithm, Rabin-Karp algorithm, testing the membership of a regular language. (3 lectures) Basic Number algorithms: Reciprocal and square algorithms to show that the time complexities of multiplication, squaring, reciprocal, and division are same. Extension to polynomials. (3 lectures) Modulo Representation of integers/polynomials: Chinese Remainder Theorem, Conversion between base-representation and modulo-representation. Extension to polynomials. Application: Interpolation problem. (3 lectures)

			Discrete Fourier Transform (DFT): In complex field, DFT in modulo ring. Fast Fourier Transform algorithm. Schonhage-Strassen Integer Multiplication algorithm. (4 lectures) Linear Programming: Geometry of the feasibility region and Simplex algorithm. (2 lectures) NP-completeness: Examples, proof of NP-hardness and NP-completeness. (2 lectures) One or more of the following topics based on time and interest: Approximation algorithms Randomized Algorithms Interior Point Method Advanced Number Theoretic Algorithm
CS603	Fundamentals Of Theoretical Computer Science		Logic: basics of propositional and first order logic, completeness & compactness results. Some applications to computer science. (E.g., theorem proving, logic programming). Theory of computation: Church's thesis, undecidability. Computational complexity: time & tape bounds, time & tape bounded simulations, notion of complexity classes, classes P & NP, NP-completeness, some natural NP-complete problems.
CS610	Programming For Performance	3-0-0-0-[9]	The course will primarily focus on the following topics: Introduction: Challenges in parallel programming, correctness and performance errors, understanding performance, performance models Exploiting spatial and temporal locality with caches, analytical cache miss analysis Compiler transformations: Dependence analysis, Loop Transformations Shared-memory programming and Pthreads Compiler vectorization: vector ISA, auto-vectorizing compiler, vector intrinsics, assembly OpenMP: Core OpenMP, Advanced OpenMP, Heterogeneous programming with OpenMP Parallel Programming Models and Patterns Intel Threading Building Blocks GPGPU programming: GPU architecture and CUDA Programming Performance bottleneck analysis: PAPI counters, Using performance analysis tools Optional Topics Heterogeneous Programming with OpenMP Fork-Join Parallelism Concurrent data structures Shared-memory synchronization Memory consistency models Transactional memory

CS614	Linux Kernal Programming		Understanding a full fledged operating system is desirable to develop new OS level functionalities in research and technology development. The goal of this course is to expose students to Linux OS (a.k.a. Linux Kernel) internals to provide an up-close view of its design and features. At the end of the course, students are expected to be confident to approach designing new OS level features when required. The course will primarily be structured around exercises, assignments and a project. Every topic will be introduced before its corresponding hands-on component. The exercises and assignments, meant to get an inside view of the kernel, will lead up to a project that will have to be submitted as the final submission for the course. For some of the concepts, recent research works proposing extensions/optimizations will also be covered. Course Objectives: 1. Understanding the design of Linux kernel components 2. Experiencing the kernel by passive/active observation 3. Extending the Linux kernel for understanding, self-satisfaction/falsification ... 4. Exploring current research trends in OS, Linux being the reference OS Prerequisites: 1. Undergraduate OS course, C programming proficiency 2. Access to a personal laptop or remote computer from the class room Syllabus 1. Introduction: OS concepts catch-up, Linux kernel overview, Extending the kernel: building a modified kernel, writing simple kernel modules 2. User-kernel interfacing: system calls, proc/sysfs, character devices, device memory maps 3. Kernel execution contexts: processes, threads, kernel threads, interrupts, bottom halves/softIRQs 4. Process management: Linux kernel scheduler, context switching, kernel synchronization 5. Memory management: Virtual memory, page cache 6. Filesystems: The VFS layer, kernel-Filesystem interfacing 7. Generic block layer: Block I/O interfacing, kernel block I/O scheduler 8. Device drivers: Device probe and sw/hw configurations, event registration, communication

			Grading In class exercises: 10% Assignments: 30% (3 to 4 assignments) Project: 30% (Group of maximum two students) Midsem/Quizzes + Endsem : 30%
CS615	Skyline Queries In Database		Skyline Queries
CS616	Human-Centered Computing		Human-centered computing(HCC) studies computing systems that are designed to support human activity. For example, search engines support information search, e-commerce supports economic consumption; increasingly, computing systems are also taking over managerial and organizational roles in service-sharing ecosystems. At the core of all such systems lie assumptions about the needs and expectations of humans, their eventual design is meant to facilitate these expectations. How specifically should a natural language search query be interpreted, based on a user's past search history? How diverse should a music playlist recommendation be, based on the current pattern of song choices of the user? The academic discipline of HCC studies such questions in the bigger theoretical structure of a two-way interaction between agent expectations and system design: systematizing various elements of human behavior that can be reliably measured by computing systems; and determining how best to design computing systems that can adaptively interact with such behavioral elements. This course offers a hands-on introduction to human-centered computing: reviewing a subset of current applications and open problems. The course comprises four modules, each one built, studio-style, around a hands-on mini-project that students will work on, individually or in groups. Theory and empirical methods will be introduced to the extent that they help the students with their projects. The course will begin with addressing topics relevant to currently mature technologies (search), transition to address currently active (recommender systems) and inchoate(affective computing) research areas and finally touch upon the common core of AI research that is the theoretical frontier in human-facing computing (goal-directed agents). Sample Course Outline Prelims Lesson 1: Intro, logistics, overview Lesson 2: Math basics (matrix operations, probability) Lesson 3: Different flavors of mathematical models Lesson 4: Model fitting, regularization Lesson 5: Programming basics (Matlab/python) Quiz

			Module 1: Search Mini-project (Topic model) Lesson 1: Classical search/information retrieval Lesson 2: Query completion Lesson 3: Contextual/topical search foci Project intro lecture Lesson 4: Information scent and other foraging models Lesson 5: Temporal information retrieval Lesson 6: Serendipity, discovery Quiz Project presentations Module 2: Recommendations Mini-project (Movielens) Lesson 1: Recommender systems Lesson 2: Collaborative filtering Lesson 3: Feature selection, SVD Project intro lecture Lesson 4: Different flavors of REs Lesson 5: Validation, measurement metrics Lesson 6: Diversity Quiz Project presentations Module 3: Emotions Mini-project (Sentiment analysis) Lesson 1: Theories and schema Lesson 2: Sentiment analysis Lesson 3: Affect measurement (computer vision, survey instruments, activity monitoring) Project intro lecture Lesson 4: Bots Lesson 5: BCI Lesson 6: Boredom/ennui Quiz Project presentations Module 4: Goals Project (Roomba) Lesson 1: Basic goal-directed agents Lesson 2: Hebbian/reinforcement learning Lesson 3: Explore-exploit dilemma Project intro lecture Lesson 4: Curiosity, perseverance, fluctuations Lesson 5: Deep principles – flow, connectedness, homeostasis, etc. Lesson 6: Gamification Quiz Project presentations
CS617	Database Queries		Optimization and evaluation of relational queries: conjunctive query optimization, optimization of queries involving union and difference operators, algorithms for performing joins. Limitations of relational algebra as a query language. Fixed-point queries and Horn-clause queries. Optimization and evaluation of Horn-clause queries:

			filtering data flow method, magic set and generalized counting methods, clause and literal deletion problems. The boundedness problem, reducing the complexity of recursion, Duplicate clause removal. Incorporating functions, sets and negations into Horn-clause queries.
CS618	Indexing And Searching Techniques In Databases		Database Queries Hashing Memory-based Index Structures Hierarchical Structures Distance Functions Distance-based Structures Curse of Dimensionality High-dimensionality Structures Dimensionality Reduction Techniques Data Representation Techniques
CS619	Advances In DBMS		User interfaces: forms, graphics, semi-graphics, spread sheet, natural language. Query optimization: techniques like query modification; Object oriented databases: notion of abstract data type, object oriented systems, object oriented db design. Expert data bases: use of rules of deduction in data bases, recursive rules. Fuzzy data bases: fuzzy set & fuzzy logic, use of fuzzy techniques to define inexact and incomplete data bases.
CS621	Topics In Contemporary Microarchitecture		Performance as well as non-performance issues in current michroarchitecture research and development. Modern techniques to fight control dependence (advanced branch predictors), and data dependence (perfecting algorithms, data speculation techniques), and techniques to scale michroarchitectures for supporting large number of in-flight instructions. Design of microprocessors for low power, reliability, and security. Power/performance trade-offs and metrics, transient fault detection and recovery, designs for reliability and hardware level security (memory integrity and code pointer protection).
CS622	Advanced Computer Architecture		Single-threaded execution, traditional microprocessors, DLP, ILP, TLP, memory wall, Parallel programming and performance issues, Shared memory multiprocessors, Synchronization, small-scale symmetric multiprocessors on a snoopy bus, cache coherence on snoopy buses, Scalable multiprocessors, Directory-based cache coherence, Interconnection network, Memory consistency models, Software distributed shared memory, multithreading in hardware, Chip multiprocessing, Current research and future trends.

CS623	VLSI Design For Parallel Architectures		Introduction to hierarchical structural design. Role of CAD in VLSI design process. Techniques and algorithms for symbolic layout and routing. CMOS processing technology, CMOS building block. Use of pipelining and parallelism, self-synchronized esigns, VLSI computing structures. Introduction to systolic arrays, mapping algorithms on systolic arrays, design of systolic arrays, system examples and design exercises.
CS624	Topics In Embedded Systems		Current topics in the design, specifications and analysis of embedded systems. The course will have the contemporary coverage of topics such as specifications of embedded systems, analysis of embedded systems, interface to the real-time operating systems, design case studies, design methodologies, etc. Other topics may include verification of embedded systems like formal verification, co-simulation, etc., estimation of hardware and software costs, partitioning, synthesis (hardware, software, memory, bus), retargetable usage of the software, specification and verification of the OS schedules, hard and soft real-time operating systems, and fault tolerant systems.
CS625	Advanced Computer Networks		Introduction: Overview of computer networks, seven-layer architecture, TCP/IP suite of protocols, etc. MAC protocols for high-speed LANS, MANs, and wireless LANs. (For example, FDDI, DQDB, HIPPI, Gigabit Ethernet, Wireless ethernet, etc.) Fast access technologies. (For example, ADSL, Cable Modem, etc.) IPv6: Why IPv6, basic protcol, extensions and options, support for QoS, security, etc., neighbour discovery, auto-configuration, routing. Changes to other protocols. Application Programming Interface for IPv6. 6bone. Mobility in networks. Mobile IP. Security related issues. IP Multicasting. Multicast routing protocols, adderss assignments, session discovery, etc. TCP extensions for high-speed networks, transaction-oriented applications. Other new options in TCP. Network security at various layers. Secure-HTTP, SSL, ESP, Authentication header, Key distribution protocols. Digital signatures, digital certificates.
CS626	Fault Tolerant Computing Systems		The course will discuss the principles & practice of fault tolerance in software and distributed systems. Some of the topics to be covered in the class are: system model - error, failure, faults, software fault tolerance, Byzantine agreement, fail-stop processors, stable storage, reliable and atomic broadcasting, process resiliency, data resiliency &

			recovery, commit protocols, reliability modeling & performance evaluation, crash recovery in databases, and voting methods.
CS627	E-Commerce		The objective of this course is to study the technologies and architectures that are in use in E-commerce today. The topics to be covered include: Supporting technologies and tools, Architecture (e.g. Java commerce solution), Protocols and standards, Security, Business models, Payment mechanisms, and Case studies.
CS628	Computer Systems Security		The course requires sufficient programming knowledge, system knowledge, and immense interest in understanding cyber security and cyber defense. So please decide whether you would like to consider this. Syllabus **Major, Measurable Learning Objectives** Having successfully completed this course, the student will be able to: Discover software bugs that pose cyber security threats, explain and recreate exploits of such bugs in realizing a cyber attack on such software, and explain how to fix the bugs to mitigate such threats Discover cyber attack scenarios to web browsers, and web servers, explain various possible exploits, recreate cyber attacks on browsers, and servers with existing bugs, and explain how to mitigate such threats Discover and explain cyber security holes in standard networking protocols, both in network architecture, standard protocols (such as TCP/IP, ARP, DNS, Ethernet, BGP etc), explain mitigation methods and revisions of standards based on cyber threats. Discover and explain mobile software bugs posing cyber security threats, explain and recreate exploits, and explain mitigation techniques. Articulate the urgent need for cyber security in critical computer systems, networks, and world wide web, and explain various threat scenarios Articulate the well known cyber attack incidents, explain the attack scenarios, and explain mitigation techniques Explain the difference between Systems Cyber Security, Network Cyber Security, and cryptography, crypto-protocols etc. Articulate the cyber threats to critical infrastructures Prerequisites And Co-Requisites Prerequisites for this course is a very strong programming background with knowledge of program run-time environment, usage of debuggers, and knowledge of shared libraries or dynamically linked libraries. Some knowledge of x86 assembly language or similar assembly language will be

			assumed. Some knowledge of Operating Systems especially memory management, virtual memory etc will be assumed. We will also assume that the student knows basic network protocols such as TCP/IP, DNS, routing etc. We will further assume that the student is familiar with a client/server architecture of the world wide web -- where browser is a client to a web server. Further, prior knowledge of a scripting language such as shell scripting, perl, python and/or Ruby will be beneficial. Knowledge of Javascript, PHP or other web programming might be very useful. Prior familiarity with preliminaries of cyber security would be helpful but not required. A quiz will be administered in the very first class in the beginning. The quiz will help you determine your standing with respect to above mentioned prior knowledge. **Texts and Special Teaching Aids** There is no specific text. We will provide all material via moodle. When the class moodle website will be up, each student should immediately register him/herself for this class on moodle. Most communications, assignments, course material will be only available via moodle. So it is extremely important that all students must be on the course moodle site. Outline Here is a tentative outline for the course -- but this is not set in stone. Some topics may be excluded, and some other topics may be included depending on the progress of the course. **Section 1: Software and System Security [30%]** 1. Control hijacking attacks – buffer overflow, integer overflow, bypassing browser memory protection 2. Sandboxing and Isolation 3. Tools and techniques for writing robust application software 4. Security vulnerability detection tools, and techniques – program analysis (static, concolic and dynamic analysis) 5. Privilege, access control, and Operating System Security 6. Exploitation techniques, and Fuzzing **Section 2: Network Security & Web Security [40%]** 1. Security Issues in TCP/IP – TCP, DNS, Routing (Topics such as basic problems of security in TCP/IP,, IPsec, BGP Security, DNS Cache poisoning etc) 2. Network Defense tools – Firewalls, Intrusion Detection, Filtering 3. DNSSec, NSec3, Distributed Firewalls, Intrusion Detection tools

			4. Threat Models, Denial of Service Attacks, DOS-proof network architecture 5. Security architecture of World Wide Web, Security Architecture of Web Servers, and Web Clients 6. Web Application Security – Cross Site Scripting Attacks, Cross Site Request Forgery, SQL Injection Attacks 7. Content Security Policies (CSP) in web 8. Session Management and User Authentication, Session Integrity 9. Https, SSL/TLS 10. Threat Modeling, Attack Surfaces, and other comprehensive approaches to network design for security **Section 3: Security in Mobile Platforms [15%]** 1. Android vs. ioS security model, threat models, information tracking, rootkits 2. Threats in mobile applications, analyzer for mobile apps to discover security vulnerabilities 3. Viruses, spywares, and keyloggers and malware detection **Section 4: Introduction to Hardware Security, Supply Chain Security [5%]** 1. Threats of Hardware Trojans and Supply Chain Security 2. Side Channel Analysis based Threats, and attacks **Section 5: Issues in Critical Infrastructure and SCADA Security [10%]** 1. Security issues in SCADA 2. IP Convergence Cyber Physical System Security threats 3. Threat models in SCADA and various protection approaches 4. Machine learning and SCADA Security Grading Semester grades will be based on the following weights: Attendance & In-Class Exercises 20% (including pop quizzes) Projects & Assignments 60% (10% each for 6 assignm projects) Midterm Exam 10% Final Exam 10% Semester grades will be determined after all work is completed and graded. Point ranges for letter grades will be based on a several factors, including absolute and relative performance. Letter grades will not be based on a fixed, predetermined curve or point range.

			Unless otherwise stated on the class **all graded assignments must be submitted by 11:55 pm on the specified due date.** There will be a 10% penalty for each 24 hour delay in submitting an assignment. If you feel that an error is made in grading an assignment or an exam, you must present a written appeal within one week after the assignment or exam is returned to you. Verbal appeals are not allowed and grades will not be changed after the one week period. Your appeal should be specific. Submit all appeals to the instructor.
CS629	Parallel Execution Of Programs		The course explores major avenues for extracting parallelism from a sequential program in addition to introducing conventional dependence analyses. The techniques include shape analysis, may-alias analysis, points-to analysis, thread-level speculation, hardware transactional memory, and promising hardware/software hybrid techniques that take help from a managed run-time system to improve parallelism such as software transactional memory, hybrid transactional memory, and hybrid analysis techniques (such as inspector/executor, dynamic parallelization, etc.).
CS630	Advanced Operating Systems For Embedded Systems, Pervasive Computing And Internet Of Things	2-0-3-0-9	The course teaches operating systems for embedded systems, mobile computers and a diversity of devices with network connectivity (internet of things). A significant component of the course are the hands-on lab assignments that will be done by the students using a state-of-the-art and open source operating system, namely Tizen (already in use in some of the Samsung smartphones: http://www.tizenphones.com/). The goal of the course is to provide a platform for students to understand and develop hands-on knowledge of advanced operating systems. The course will be primarily taught by researchers from Samsung Research India. Tizen is an open source OS from Samsung. The course has a significant project component. A list of projects will be provided to the students. There are multiple tracks in the course, including, Multimedia, Network and Connectivity+Bluetooth, Web framework, Graphics etc.. Students will take a project in one of these tracks. Experts in each track will also come to guide the students from time to time. The course is meant for those who are seriously interested in advanced OS and have strong hands-on grasp of operating systems at the level of CS330 or equivalent. Topics: (Given on the next page)
CS631	Cyber Security Of Critical Infrastructures		Prerequisites And Co-Requisites Prerequisites will include at least one course in operating systems, and one course in networking.

Prior familiarity with preliminaries of cyber security would be helpful but not required. With the instructor's permission, one or both prerequisites can be waived provided the instructor feels that the student has adequate exposure to the relevant topics in those courses.

Major, Measurable Learning Objectives

Having successfully completed this course, the student will be able to:

Identify the key research questions in the area of cyber-security of critical infrastructure

Apply research methods which includes survey, experiments, and articulation of research problems in this area, and methods for finding solutions to selected problems

Present in written and/or verbal form key findings in the specific subject area of the course from contemporary research papers.

Read and analyze research papers from journals and conferences in the specific subject area of the course.

Syllabus

The students will be exposed to the following topics:

Stuxnet worm and its after effects in the Critical Infrastructure security

Consequent Presidential Executive Order for Securing Critical Infrastructure in 2013 and its impact: Policy Issues in Security of Critical Infrastructure

Security and Vulnerability of Cyber-Physical Infrastructures

Game Theory and other analytical modeling of the security problems of critical infrastructures

Security of the Networked Infrastructure

Event monitoring, Event Correlation, and Situational Awareness

Case Studies – Smart Grid, Smart Infrastructure etc.

Vulnerability Database and its importance

The course will consist of instructor presentations, student presentations, guest lectures, and group discussions. This course will be quite research focused, and the goal of the course will be to enable students to find research topics in the domain of cyber-security of critical infrastructure.

Module	Topic	No. of Lectures
Introduction	**Critical Infrastructures such as Power Grid, Railways Systems, Transportation Systems,**	**4**

				Water/Sewage Systems and their automation architecture, Vulnerabilities, and Past Cases of Cyber Security Compromises and Trends Stuxnet Case Study, and Reaction through US Presidential Executive Order	
			Industry Automation and SCADA Systems	SCADA Based Control, Sensors (IEDs, PLCs), field network and its protocols (profibus, DNP3 etc)	8
			Modeling SCADA Systems as a Cyber Physical System Model	Cyber Physical System Modeling, Plant Models, Feed Back Control Model, and Anomaly Detection Models	2
			Cyber Threat Modeling	Various Types of Cyber Threats to Industrial Critical System Modeled in a 3 dimensional Attack Space in terms of adversary Model and Understanding various attacks in this Model	6
			Cyber Threat Mitigation	Various Techniques to mitigate various attacks such as replay attack, zero-dynamics attack, stealthy attacks etc	4
			Virtual SCADA Simulation Platform	Virtual SCADA Simulation Platform to be used in Projects and Homeworks, Its architecture, implementation, and instruction on installation	4
			Machine Learning Techniques	Cyber Physical Systems under attacks and study of their physical dynamics to distinguish between a normal behavior vs. behavior under attack, use of machine learning techniques to distinguish and detect in real-time	10
			Game Theoretic formulation	Modeling an attacker vs. Defender game, Nash Equilibrium criteria, and	4

<table>
<tr><td></td><td></td><td></td><td>
<table>
<tr><td></td><td>understanding advantages of game theoretic modeling</td><td></td></tr>
</table>
Grading
Semester grades will be based on the following weights.
In-Class Exercises : 10% (Based on participation as described below)
At-Home Exercises: 40% (10% each for 4 at-home exercises)
Midterm Exam: 10%
Final Exam: 10%
Projects: 30% (Number of Projects yet to be determined)
</td></tr>
<tr><td>CS632</td><td>Topics In Distributed Systems</td><td></td><td>Local area networks, concurrency control and recovery, distributed languages and communication primitives, file servers, case studies of distributed systems.</td></tr>
<tr><td>CS633</td><td>Parallel Computing</td><td>3-0-0-9</td><td>
<table>
<tr><th>S. No.</th><th>Broad Title</th><th>Topics</th><th>No. of Lectures</th></tr>
<tr><td>1.</td><td>Introduction</td><td>Why parallel computing? Shared memory and distributed memory parallelism, Amdahl’s law, speedup and efficiency, supercomputers.</td><td>2</td></tr>
<tr><td>2.</td><td>Message passing</td><td>MPI basics, point-to-point communication, collective communication, synchronous/asynchronous send/recv, algorithms for gather, scatter, broadcast, reduce.</td><td>8</td></tr>
<tr><td>3.</td><td>Parallel communication</td><td>Network topologies, network evaluation metrics, communication cost, routing ininterconnection networks, static and adaptive routing, process-to-processor mapping.</td><td>6</td></tr>
<tr><td>4.</td><td>Performance</td><td>Scalability, benchmarking, performance modeling, impact of network topologies,</td><td>7</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td>
<table>
<tr><td></td><td></td><td>parallel code analysis and profiling.</td><td></td></tr>
<tr><td>5.</td><td>Designing parallel codes</td><td>Domain decomposition, communication-to-computation ratio, load balancing, adaptivity, case studies: weather and material simulation codes.</td><td>7</td></tr>
<tr><td>6.</td><td>Parallel I/O</td><td>MPI I/O algorithms, contemporary large-scale I/O architecture, I/O bottlenecks.</td><td>6</td></tr>
<tr><td>7.</td><td>Additional topics</td><td>Job scheduling, RDMA, one-sided communication, NVM, extreme scale computing: issues and trends.</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>CS634</td><td>Mobile Computing</td><td></td><td>Introduction: Challenges in mobile computing, coping with uncertainities, resource poorness, banwidth, etc. Cellular architecture, co-channel interference, frequency reuse, capacity increase by cell splitting. Evolution of mobile system: CDMA, FDMA, TDMA, GSM.
Mobility Management: Cellular architecture, Co-channel interference, Mobility: handoff, types of handoffs; location management, HLR-VLR scheme, hierarchical scheme, predictive location management schemes. Mobile IP, cellular IP.
Publishing & Accessing Data in Air: Pull and push based data delivery models, data dissemination by broadcast, broadcast disks, directory service in air, energy efficient indexing scheme for push based data delivery.
File System Support for Mobility: Distributed file sharing for mobility support, Coda and other storage manager for mobility support
Ad hoc Network Routing Protocols: Ad hoc network routing protocols, destination sequenced distance vector algorithm, cluster based gateway switch routing, global state routing, fish-eye state routing, dynamic source routing, ad hoc on-demand routing, location aided routing, zonal routing algorithm.
Mobile Transaction and Commerce: Models for mobile transaction. Kangaroo and joey transactions, team transaction. Recovery model for mobile transactions. Electronic payment and protocols for mobile commerce.</td></tr>
<tr><td>CS635</td><td></td><td></td><td></td></tr>
</table>

CS636	Analysis Of Concurrent Programs		Programming Language Basics: Syntax, Semantics, Types Review of concurrent programming paradigms: shared-memory, message-passing, partitioned global address space Synchronization primitives: locks, monitors, semaphores, flags, barriers, condition variables Concurrency bugs: data races, race conditions, atomicity violations, deadlocks. livelocks Consistency models: strict consistency, sequential consistency, linearizability (atomic consistency), relaxed memory models, memory fences Dataflow analysis for concurrent programs Deductive Verification: Hoare Logic, Owicki-Gries, Rely-Guarantee, Concurrent Separation Logic State-space reduction techniques: Formal modeling of a concurrent system, Mazurkiewicz traces, Partial-Order reduction techniques (persistent sets, sleep sets), dynamic partial-order reduction Dynamic Analysis: Fuzzing, probabilistic concurrency testing, statistical analysis Bounded model-checking for concurrent programs Analysis of message-passing programs
CS637	Embedded And Cyber-Physical Systems		Modeling Dynamic Behaviors and Control: Continuous Dynamics, Feedback Control, Discrete Systems, Hybrid Systems, Composition of State Machines, Concurrent Models of Computation Design and Implementation: Sensors and Actuators, Embedded Processors, Memory Architectures, Input and Output Interface, Multitasking, Scheduling Analysis and Verification: Invariants and Temporal Logic, Equivalence and Refinement, Rechability Analysis, Model Checking, Timing Analysis
CS638	Formal Methods In Robotics And Automation	3-0-0-9	Basics of Verification: Finite State Machines, Linear Temporal Logic (LTL), Computation Tree Logic (CTL), Automata-Based LTL Model Checking, μ-Calculus Model Checking, Markov Decision Process, Probabilistic Computation Tree Logic (PCTL), Probabilistic Model Checking, Bisimulation Equivalence, Reactive Synthesis, Binary Decision Diagram, SAT and SMT Solvers Control Theory: Basics of Feedback Control Theory, Hybrid Systems, Discrete Abstraction of Hybrid Systems Motion Planning: Basics of Motion Planning, Sampling-Based Motion Planning, Feedback Motion Planning, Multi-Robot Motion Planning Formal Methods for Robotics: Motion Planning from LTL and μ-Calculus Specification, Motion Planning from Temporal Logic Specifications with Probabilistic Satisfaction Guarantees, Reactive Motion Plan Synthesis, Explaining Unsynthesizability for High-Level Robot Behaviors,

			Multi-Robot Motion Planning using SMT Solvers, Software Synthesis for Semi-Autonomous Systems
CS639	Program Analysis, Verification and Testing		The long yearned dream of establishing the correctness of programs has cata-pulted program analysis, verification and testing into the league of the one of the most exciting research areas. This course aims to impart the students a grasp of the theoretical fundamentals of the subject; alongside, it also intends to provide them a purview of the practise via the study of some of the modern verification and testing tools. Following is the outline of the proposed course: Dataflow Analysis; Interprocedural Analysis: functional, call-string and graph reachability based approaches Abstract Interpretation; Weakest Precondition, Floyd-Hoare Logic, Separation Logic; SoftwareModel Checking: symbolic execution, state-space reduction, state-less model checking, counter-example guided abstraction refinement, model checking of concurrent programs Program Testing: program testing basics, automatic test-case generation, directed testing.
CS640	Computational Complexity		Complexity Classes. NP and co-NP, Results on the structure of NP-complete sets, Sparse NP-hard sets, Basic Inclusions and Separations, Nondeterministic Space Classes, Logarithmic Space, A PSPACE complete problem, Polynomial Hierarchy, PH through Alternating Quantifiers, Universal Relations, Probabilistic Classes, Schwartz-Zippel Lemma and BPP, BPP and its relationship with other Complexity Classes.
CS641	Modern Cryptology		Basics of finite fields. Private and Public-key cryptography, existing cryptosystems and their security. Cryptanalysis of existing systems. Zero-knowledge protocols, One-way functions. Advanced protocols for different applications, e.g. e-cheque, e-cash etc. Network and System level security issues.
CS642	Circuit Complexity Theory		The course aims at a comprehensive overview of results on the circuit complexity classes and their relationship with the Turing based classes. The topics to be covered in the course are as follows: The class NC and its properties; Charecterization of class P by circuits The classes DLOG, NLOG, LogCFL and their properties The class SC, proof of the relationship RL is a subset of SC The class NC1 and its charecteriztions

			The class TC0 and its charecterizations The class ACC and its charecterizations The class AC0 and its charecterizations Lower bounds for AC0, for AC0[m] where_m_is a prime power and for TC02.
CS643	Abstract State Machines		Examples of sequential abstract state machines (ASMs, for short) specifying some familiar algorithms. Proof of sequential ASM thesis which states that all sequential algorithms can be captured by sequential ASMs. Computations with abstract structures, choiceless polynomial time. ASM specification of parallel and distributed algorithms. ASM methodology for specifying semantics of programming languages, and for verification. Comparison of ASM approach with other existing methodologies for specification and verification. ASM defined fine complexity classes. ASMs and meta-finite models.
CS644	Finite Automata On Infinite Inputs		Finite automata on infinite words and trees: Complementation, determinization and algorithms for checking emptiness. Connections with logic: Finite automata and monadic second order (MSO) logic on words and trees. Decidability of MSO theory of various infinite graphs, methods of interpretation and unfolding. Applications: Decision procedures for temporal logics. Modelling, verification and synthesis of systems. Effective theory of infinite games. Miscellaneous : Timed and hybrid automata. Probabilistic transition systems. Visual pushdown automata.
CS645	Topics In Design And Analysis Of Algorithms		Introduction. Linear time special cases for Disjoint set Union-Find algorithms. Topics in Computational Geometry like Selection algorithms and application to convex hull (ultimate convex hull algorithm), linear programming in two and three dimensions. Topics in Algorithmic Graph Theory like Planar graph separators. Integer sorting and improved algorithms for shortest paths and minimum spanning tree (general and integer weights). Polynomial time algorithms for matching and minimum cost network flow problems. Scaling algorithms for network flow problems.
CS646	Parallel Algorithms		Complexity measure for a parallel algorithms. Parallel searching algorithms: maximum/minimum, median, K-th largest/smallest element. Parallel sorting algorithms. Parallel graph algorithms: parallel graph search &, tree traversal algorithms, parallel algorithms for

			connectivity problems, parallel algorithms for path problems.
CS647	Advanced Topics In Algorithms And Data Structures		The course intends to deal with advanced aspects of algorithm: design and analysis including data structures, analysis and lower bound proofs, amortized complexity of algorithms. Fibonacci heaps and self-adjusting search trees, Splay trees, linking and cutting trees. State-of-the-art algorithms for minimum spanning trees, shortest path problem. Network flows -- preflow-push algorithms, max flow algorithm, and scaling algorithms. Matching, blossoms, Micali-Vazirani algorithm. Lower bound theory for parallel computations.
CS648	Randomised Algorithms		Review of discrete probability; Notion of randomized algorithms, motivating examples; Markov, Chebyshev inequalities, Chernoff bounds; Probabilistic method; Hashing, fingerprinting; Random walks and Markov chains. Program checkers; Polynomial identities; Randomized complexity classes, Probabilistically checkable proofs; some number theoretic problems; Approximate counting.
CS649	Logic In Computer Science		The aim of this course will be to provide introduction to some applications of logic in computer science. Following are some possible topics. At least three of these will be covered in some detail. Actual choice of topics, including depth and breadth, will depend on interest/background of the class. Communication and Concurrency: Processes as transition systems, operations on these processes (composition, hiding etc.). Bisimulation and observational equivalences. Calculus of mobile systems: pi-calculus. Some theory related to pi calculus. Logics to reason about transition systems, LTL, CTL* and modal mu calculus. Reasoning about Knowledge: Knowledge as modality, axioms of knowledge. Common knowledge, distributed agents exchanging messages, agreeing to disagree. Logical omniscience. Finite Model Theory: Expressiveness of FO and its extensions on finite structures. Games for lower bounds. Connections with complexity classes, role of order on the domain. Feasible Proofs: Propositional proof systems for tautologies. Simulation and lower bounds on length of proofs for specific systems (e.g. PHP requires superpolynomial length using resolution). Theories of weak arithmetic, provably total functions and relations to complexity theory. Full Abstraction problem for PCF: PCF as an extension of lambda calculus. Operational and denotational semantics and the full abstraction

			problem. Solutions to the full abstraction problem. Games semantics.
CS650	Topics In Lambda Calculus		Optimal reductions in lambda calculus: Levy's formalization of the problem, Lamping's algorithm and its correctness. Connections to linear logic and geometry of interaction. Inherent complexity of implementing optimal reductions. Categorical semantics of lambda calculus: Introduction to category theory. Cartesian closed categories and typed lambda calculus. Relation to deductive systems. Categories with reflexive elements, a construction by D. Scott. Games semantics: Games semantics for lambda calculus. Solution to full abstraction problem via games. PCF, an extension of typed lambda calculus.
CS651	Concurrent Data Structures And Algorithms		The parallelizing compilers have come a long way in helping the programmers automatically transform sequential programs into parallel ones. However, a compiler can only restructure a program and cannot change the underlying algorithm. As a result, parallelizability of a particular sequential program ultimately depends on how the computation is expressed. This course will emphasize the well-known fact that every computation, after all, is a composition of the data structures used and the algorithms supported on those data structures. The algorithms supported by the data structures greatly influence the chances of parallelizing the computation. This course is divided into three parts. The first part will develop some of the general tools needed to design and argue about the correctness of concurrent algorithms. These include the concepts of linearizability, compositional linearizability, non-blocking and blocking synchronization on shared memory, and memory consistency models. The second part of the course will focus on developing efficient parallel algorithms for some of the frequently used data structures such as linked lists, queues, stacks, hash tables, skiplists, priority queues, etc. The primary emphasis will be on minimizing synchronization, often resorting to optimistic parallelization via dynamic conflict detection with the help of checkpoints. The third part of the course will establish the connection of transactional computation with these concurrent algorithms and present a brief overview of software and hardware implementation of transactional computation.
CS652	Computer Aided Verification	3-0-0-0-9	Pre-requisites: The course does not have any formal prerequisites. The students are expected to have

			mathematical maturity of the level of an undergraduate degree in engineering. However, some fa- miliarity with _nite state machines and theory of computation, and programming experience will be helpful. Proposed by: Indranil Saha Estimated Enrollment: 30 Other faculty members who could be interested in teaching the course: Prof. Amey Karkare (CSE), Prof. Subhajit Roy (CSE), Prof. Anil Seth (CSE), Prof. Sandeep K. Shukla (CSE), Prof. Sunil Simon (CSE) and others Departments which may be interested: Electrical Engineering, Mechanical Engineering, Aerospace En- gineering, Mathematics Level of the course: PG (6xx level). Short Description Correctness of the hardware, software and cyber-physical systems is of prime importance to prevent _nancial loss or loss of life. With the increase in the complexity of the systems, ensuring the correctness of the systems becomes a signi_cant challenge and entails algorithmic methods that can automatically ensure correctness of the systems without manual intervention. This course will discuss the mechanisms for modeling a system and capturing its requirements formally, and algorithms and techniques for verifying the model with respect to its formal speci_cation. The course will also look into various software tools that have been successful in utilizing the algorithmic techniques to solve practical veri_cation problem. Topics Modeling of Systems: Modeling of concurrent systems, timed systems, hybrid systems and probabilistic systems. Speci_cation Languages: Linear time properties, regular properties, Linear Temporal Logic (LTL), Com- putation Tree Logic (CTL), Timed Computation Tree Logic (TCTL), Probabilistic Computational Tree Logic (PCTL)

			Techniques for Model Checking: Explicit-State Model Checking, Symbolic Model Checking, Bounded Model Checking, Equivalence and Abstraction, Partial Order Reduction BDD, SAT and SMT: Binary Decision Diagrams, Satis_ability Solvers, Satis_ability Modulo Theories (SMT) Solvers. Software Tools: Popular formal methods tools such as Spin, NuSMV, SAL, UPPAAL, SpaceX, Prism, Z3 and CUDD.
CS653	Functional Programming		**ML** (CAML dialect) or **Haskell** or some other functional language; **lambda**-calculus and combinators; abstraction and higher-order functions; lazy and eager evaluation; types, polymorphism, and type inference; Equations and pattern matching; SECD machine/G machine; denotational semantics of functional languages; implementing functional languages.
CS654	Software Architecture		Complex software systems require abstraction and analysis at an architectural level of abstraction. In this course we study, typical software system structures (architectural styles), techniques for designing and implementing these structures, models for characterizing and reasoning about architectures, and tools architectural modelling. Role of architecture in Software engineering; Enterprise Architectures, Zachman's Framework; Architectural Styles, Design Patterns; Architecture Description Languages; Product-line architectures; Component based development
CS655	Topics In Linear Programming	3-0-0-0-4 (CS655A, PG), 3-0-0-0-9 (CS655B, UG)	The course will focus on basics of convexity theory, linear programming and application of linear programming in computer science. There will be broadly three parts of the course interrelated with each other. The first part will cover the basics of convex sets, cones and polyhedras; essential to understand the structure behind convex optimization and motivate why so many special cases of it can be solved efficiently. This will also cover hyperplane separation theorems. Linear programming will be introduced and duality theory will be covered in detail. The second part will talk about major algorithms which are used today to solve linear programming. The major three algorithms, simplex, ellipsoid and interior point method will be discussed. As an assignment they will have to convert a problem into a linear program and solve it using the existing linear programming solvers.

			The final part will cover some applications in computer science. Specifically we will cover the multiplicative update method and a recent result where linear programming gives best approximation algorithm for a set of constraint satisfaction problems. Applications in network flows will also be covered.
CS656	Algorithmic Game Theory	3-0-0-9	Basics of utility theory - the von Neumann-Morgenstern axioms for utility theory and characterisation theorem. Definition of strategic form games, examples. Some fundamental types of solutions concepts based on dominance (strict, weak), stability (equilibrium) and security (maximin). Relationship between these solution concepts in general games and in the special case of two player zero sum games. Mixed extension of a strategic form game. Solution concepts based on mixed strategies. Nash's Theorem. Complexity of computation of Nash equilibrium. Special case - 2 player zero sum games, correspondence between strong duality in LP and minimax theorem. Improvement dynamics and potential games. Monderer-Shapley characterisation. Congestion games Computation of pure strategy equilibria in potential games - PLS completeness Graphical games. 0/1 polymatrix games and hardness of computing equilibria. Some subclasses of games which allow efficient computation of equilibria. Efficiency of equilibria (price of stability, price of anarchy) Introduction to auction theory, first price auction, second price auction, incentive compatibility. Introduction to mechanism design, Groves Mechanism, pivotal mechanism and Bailey Cavallo mechanism. Green-Laffont result. Stable matching. Matching markets and sponsored search.
CS657	Information Retrieval		Search engines have become ubiquitous with modern information needs. What is it that enables finding a document in the blink of an eye from a seemingly unending catalogue of documents, namely, the Internet? The answer is "information retrieval", essentially defined as the retrieval of information (mostly text) efficiently from a large collection of objects (mostly documents). In recent years, information retrieval needs have expanded to music, image, videos, graphs, and so on.

			This course will cover the basic methods of information retrieval. In particular, it will cover the entire pipeline of building an information retrieval system, starting from the basic boolean retrieval model to designing web-scale engines. Emphasis will also be given on the recent trends in the field.
			The tentative topics to be discussed are:
			
			1. Motivation for information retrieval
			2. Basic document retrieval
			  1. Inverted index
			  2. Querying using inverted index
			3. Tokenization
			  1. Word segmentation
			  2. Stopwords
			  3. Stemming
			4. Document scoring
			  1. Zone scoring
			  2. Term Frequency
			  3. Inverse Document Frequency
			  4. Tf-idf
			5. Document as a vector
			  1. Vector model
			  2. Document similarity
			  3. Document vector models
			  1. LDA
			  2. GLoVe
			  3. Word2Vec
			6. Scalability
			  1. Skip list
			  2. Champion list
			  3. Tiered index
			7. IR as a system
			8. IR for non-documents
			  1. Images
			  2. Graphs
			  3. Audio

CS658A	Malware Analysis And Intrusion Detection	3-0-0-0- [9]	A recent report by the IDC for smartphone operating system global market share shows that in the 3rd quarter of the year 2018, the total market share of Android was 86.8%. In May 2019, Google revealed that there are now more than two and half billion Android devices that are being used actively in a month. With the increase in popularity of Android, the number of active users and the day to day activity of each user on Android devices have also increased a lot. *This allows malware authors to target Android devices more and more*. It is reported by Gadgets360 that 8400 new instances of Android malware are found every day. *This implies that a new malware surfaces every 10 seconds.* Malware is one of the serious cyber threats which evolve daily, and can disrupt various sectors like online banking,

			social networking, etc. According to the reports published by AV-Test Institute, across various platforms – android, windows, Linux etc., there has been a tremendous growth in the number of malicious samples registering over 250,000 new malicious samples every day. *Analyzing these samples manually using reverse engineering and disassembly is a tedious and cumbersome task.* It is therefore not convenient for the security analysts. Thus, there is a dire need for automated malware analysis systems which produce effective results with minimal human intervention. Antivirus systems use the most common and primitive approach, which involves the generation of signatures of known malware beforehand and then comparing newly downloaded executables against these signatures to predict its nature. This technique drastically fails in case of any zero-day malware, a malware which has been newly created and thus a signature is not available. Other common techniques are static analysis and dynamic analysis. Static analysis analyzes the executables without executing it and predicts the results. It is generally used because it's relatively fast but fails if the malware is packed, encrypted or obfuscated. To overcome the limitations of static approaches another approach, i.e., dynamic analysis is used. It involves collecting behavioural data by executing the sample in a sandboxed environment and then using it for detection and classification. The dynamic analysis also has some limitations such as the detection of virtual environment and code coverage issues. As a result, researchers have started using the combination of both the approaches known as a hybrid approach. Intrusion into IT networks of organizations, and OT network of utilities, factories, power generation stations is another growing cyber threat that one has to cope with. Intrusion detection is the technique for analyzing various signals (network traffic, variations in CPU activities, variations in sensor data, variations in attempts to gain access etc.) in order to ascertain if an intrusion attempt is on-going or if the intrusion is taking place. Intrusion detection also can be done by rule-based techniques, signature-based techniques or anomaly detection. For those wanting to pursue a career in Cyber Security – knowing the techniques for malware analysis, and intrusion detection, ability to develop tools to carry out such analyses is very important, and that is why this course is being offered as an advanced topics course.

On completion of this course, a student should be able to: (i) Explain the vast scope of the malware borne cyber-attacks, various malware types, and platform-specific variations of malware; (ii) Explain the threat models associated with network and host intrusion by cyber-attackers; (iii) Explain the basic signs of malware infection and signs of intrusion from a security analyst's point of view; (iv) Explain various machine learning techniques and tools used for malware analysis, anomaly detection and techniques such as memory forensics; (v) Implement tools for malware analysis employing machine learning tools and libraries and measure the efficacy of their tools on labelled and unlabeled data; (vi) Implement intrusion detection tools with machine learning libraries and measure the efficacies of the tools; (vii) Read and explain most recent publications in top conferences in the field of cyber security pertaining to machine learning and intrusion detection; (viii) prepare for further research in malware analysis and intrusion detection.

The reason the two topics (a) malware analysis and classification; and (b) intrusion detection are put together in the same advanced topics course is because they share many common machine learning based techniques.

Module	Topic	No. of 1 hour Lectures
Introduction	Malware classification, types, and platform specific issues with malware, Intrusion into IT and operational network (OT) and their signs	**3**
Basic Malware Analysis	Manual Malware Infection analysis, signature based malware detection and classification – pros and cons, and need for machine learning based techniques	**5**
Advanced Techniques Malware Analysis	Static Analysis, Dynamic Analysis and Hybrid Analysis of Windows Malware, Linux Malware and Android Malware	**8**

Case Studies	Study papers in Malware Analysis from most recent conferences, Presentations and Discussions, and Implementations	**6**
Basic Intrusion Detection	Intrusion into network – Firewalls, Rule based techniques, signature based Techniques, Simple Machine Learning Models on Network Data	**4**
Advanced Intrusion Detection	Advanced Machine Learning Models for Intrusion Detection in IT Networks, Machine Learning in OT network especially with Cyber Physical Systems	**6**
Case Studies	Latest Papers in Intrusion Detection, Their theory and Implementations, and Data Analysis Techniques	**8**
Total Lecture hours		**40 hours**

Text:

There is no textbook for such a course yet. Research Papers will be the main sources of study material.

There will be other resources put on the web by the instructor.

Lecture notes, assignments, supplemental readings, and other resources will be provided via the course website

The course will consist of 3 hours of lectures per week, projects and homework, and possibly a course project.

Grading

Semester grades will be based on the following weights:

Attendance & In-Class Exercises	10%	(including pop quizzes)
Projects & Assignments	50%	(10% each for 7 assignments and projects)
Midterm Exam Final Exam	20% 20%	

			Semester grades will be determined after all work is completed and graded. Point ranges for letter grades will be based on a several factors, including absolute and relative performance. Letter grades will not be based on a curve or point range. Unless otherwise stated on the class **all graded assignments must be submitted by 11:55 pm on the specified due date via course site on canvas.** There will be a 10% penalty for each 24-hour delay in submitting an assignment. If you feel that an error is made in grading an assignment or an exam, you must present a written appeal within one week after the assignment or exam is returned to you. Verbal appeals are not allowed, and grades will not be changed after the one-week period. Your appeal should be specific. Submit all appeals to the instructor.
CS659	Autonomous Cyber-Physical Systems		The purpose of this course will be to educate undergraduate and graduate students on state-of-the-art techniques in autonomous systems from both a theoretical and practical perspective. The key difference in this course and other courses taught in robotics, artificial intelligence, machine learning and control will be that it will eschew a purely practical focus that many of the other courses favor. It will instead teach students to reason about aspects such as safety, and reliability for autonomous systems using tools from control theory, formal methods, automata theory, artificial intelligence, and logic. The Course Will Cover The Following Topics - Formal modelling and specification for CPS models - Model-based verification and testing - Various ingredients for autonomy based on AI techniques such as path planning, reinforcement learning - Basics of the software stack for autonomous systems such as sensing, perception, communication, and feedback control
CS660	Fundamentals Of Interactive Computer Graphics		Overview of the programmer's model of interactive graphics. Computer graphics and image processing and techniques. Implementation of a simple graphics package. Geometric transformations, viewing transformations, advanced display architecture. Raster algorithms and software. Techniques of visual realism.

			Algorithms for hidden edge and surface removal. Shading models, colour displays and concepts of shadows.
CS661	Big Data Visual Analytics	**3-0-0-0 (9)**	Introduction to visual analytics Data Different types of data Big data and its characteristics Foundations of data visualization Visual perception Information analysis and visual variables Data and task abstraction Software Overview of available visualization software ParaView, VTK, D3.js Scientific visualization Scientific data models Basic visualization techniques Information visualization Techniques such as Clustering, Dimension reduction, PCP, MDS, SPLOM etc. High dimensional and graph data visualization Techniques for big data visual analytics Data compression Statistical methods Information theory for big data visualization High performance algorithms for visualization Machine/Deep learning techniques for big data visualization Data exploration at extreme-scale Exascale computing In situ visual analysis Future paradigm in extreme-scale data visualization
CS662A	Introduction To Linear Logic	3-0-0-9	CourseObjective: Linear Logic was introduced by J. Y. Girard in 1987. This was followed by rapid new development of this field and its applications for a decade or so. Currently concepts and techniques related to these works are used in many areas of logic and semantics of computation. Objective of this course is to cover a set of core topics in linear logic to provide firm background to study more advanced topics. Syllabus: Sequent Calculus presentation of propositional linear logic. Cut-Elimination theorem. Undecidability. Connection with computation via Curry-Howard isomorphism. Computational interpretaions. Connection with computation via linear logic programming. Uniform proofs, an example programming language like Lolli. Categorical framework for semantics of linear logic. Games and other models.

			Proof nets. Depending on availaility of time, basics of some topics like Geometry of interaction, Abstract machines, Bounded linear logic may be taken up.
CS663	Computational Geometry		Historical perspective: complexity notions in classical geometry. Towards computational geometry, geometric preliminaries, models of computation. Geometric searching: point location problems, location of a point in a plannar subdivision, the slab method, the chain method, range - searching problems. Convex hulls: problem statement and lower bounds. Graham's scan, Jarvis's march, quick hull technique, convex hulls in two and higher dimensions, extension and applications. Proximity: divide and conquer approach, locus approach; the Voronoi diagram, lower bounds, variants and generalizations. Intersections, hidden-line and hidden surface problem. The geometry of rectangles: application of the geometry of rectangles, measure and perimeter of a union of rectangles, intersection of rectangles and related problems.
CS664	IoT System Design		Internet of Things (IoT) has gained prominence with the ever increasing connected devices, sensor systems and capability of computing resources. Thanks to the advancement of fabrication technology which has now made IoT devices and systems integral part of our daily life. An IoT system typically comprises of smart sensor nodes to collect data either real-time or offline, data communication over a network and the back-end data management & processing to extract intelligent information. The typical use cases of IoT are wearables, smart homes, smart vehicles, traffic prediction & control, weather monitoring & forecasting, indoor location-based services, health monitoring of machines & structures, augmented/virtual reality etc. Consumers and industries are the beneficiaries of such applications. Course Objectives: The focus of this introductory course would be “the smart sensor node” with emphasis on design, requirement, data interfacing and capabilities. The course would cover engineering fundamentals, blended with good industrial practices, which lead to the first-time success of the design and development of sensor node. API development, cloud computing, and data analysis would also be covered in brief. Lab sessions and case studies will supplement the classroom interactions.

			After completing this course, students will be in a position to understand various building blocks and working of state-of-the-art IoT systems. Students would also gain enough insights to conceive and build IoT systems on their own.
CS665	Secure Memory Systems		Course Objective Memory subsystem is an important layer in the computing system that has to be efficient for the whole system to operate efficiently. In the current era of computation, multiple cores are deployed in devices that range from smart-phones, laptops, desktops, servers, and cloud based systems. Though, innovations in the world of hardware and computer architecture have resulted in faster computation in terms of better performance, a lot of sensitive data that is stored and processed by these devices can get leaked through various hardware components, such as branch predictors, caches, Translation look-aside buffers (TLBs), page tables, prefetchers, Dynamic Random Access Memory (DRAM) controllers, DRAM, and non-volatile memories (NVMs). Basically, these hardware components become side-channels and/or covert-channels and become source of information leakage in the form of side-channel and covert-channel attacks (for example, the recent meltdown and spectre attacks). The goal of the course is to make students understand the various sources of attacks and their mitigation techniques at the memory systems, and design secure memory systems. The course will be a fusion of fundamentals and state-of-the-art research on secure memory systems. Course Contents **10K feet view:** Spying on passwords through memory systems The course comprises of four main modules apart from a module on preliminaries. **Module 0:** Preliminaries on Caches, DRAM, and Virtual memory systems **Module 1:** Secure Caches Side-channel and covert-channel attacks at different levels of cache hierarchy Cache attack mitigation techniques Trade-off between system performance, power, and security **Module 2:** Secure DRAMs Side-channel and covert-channel attacks at the DRAM controllers Side-channel and covert-channel attacks at the DRAM chips Attack mitigation techniques **Module 3:** Secure Virtual Memory Systems

<table>
<tr><td></td><td></td><td></td><td>Side/covert-channel attacks at the TLBs, MMU caches, Page-table walkers
Mitigation techniques at different levels of virtual memory system
Module 4: Other Topics
Reverse engineering memory systems
Interface between secure memory system, secure processor, and secure OS. Intel SGX, ORAMs
Security issues in NVMs
Guest Lectures
Clementine Maurice
Vinod Ganapathy</td></tr>
<tr><td>CS666</td><td>Hardware Security For Internet-Of-Things</td><td>3-0-0-0 (9)</td><td>Prerequisites:
Knowledge equivalent of CS220 and CS641

Who Can Take The Course:
PhD, Masters, 3rd and 4th year UG Students

Departments That May Be Interested:
CSE, EE

Course Objective
The domain of hardware security mainly covers the protection of physical device from different security threats posed by information leakage through covert channels, Trojan insertion, machine learning attacks, in- vasive or semi-invasion attacks etc. Even, secure deployment of various hardware security primitives in the untrusted environments entails several hardware and protocol level challenges. In this course, we will focus on the ever-increasing number of connected devices in IoT framework and analyse the impact of real world threats. And then, we will intro- duce various hardware security primitives for authentication and secure communication. The contents selected for the course are based on research papers from top-tier journals and conferences such as IEEE TIFS, IACR TCHES, IEEE TDSC, ACM TECS, CCS, S&P, USENIX, DAC, DATE etc. covering advanced topics of hardware security.

Course Contents

<table>
<tr><th>S. No.</th><th>Broad Title</th><th>Topics</th><th>No. of Lectures</th></tr>
<tr><td>1</td><td>IoT Building Blocks</td><td>IoT Technology Stack, Protocols, Applications, Role of Hardware</td><td>1</td></tr>
<tr><td>2</td><td>Symmetric Key Cryptography</td><td>Design Principles of AES, PRESENT and SIMON</td><td>2</td></tr>
</table>
</td></tr>
</table>

			3	Asymmetric Key Cryptography	Design Principles of RSA and ECC	2
			4	Hardware Design	Motivation, Advantages, Usecase: AES	1
			5	Power Attacks	Power Models, Differential and Correlation Power Attacks, Countermeasures	4
			6	Fault Attacks	Differential Fault Analysis, Fault Models, Countermeasures	4
			7	Timing Side Channel	Timing Attacks, Cache Attacks, Micro-Architectural Attacks, Impact on IoT	2
			8	Side Channel on Smart Devices	Use cases: Smart Light, Smart Home, Mobile App, Wearables	1
			9	Physically Unclonable Functions	Design Principles, Compositions, Machine Learning Attacks, Side Channel Attacks	6
			10	True Random Number Generator	Design Principles, NIST and AIS Test, Attacks on TRNG	3
			11	Hardware Trojan	Impact, Design methodologies, Detection Techniques	1
			12	Security Protocols	Basic attack notions of protocols, Use cases: Attacks on WPA2 handshake, Keyless entry systems of automotive system, logjam attacks in TLS Layer	2
			13	PUF Based Authentication	Authentication Protocols, Applications, Attacks, Bit commitment and oblivious transfer protocols	6
			14	Remote Attestation	Difference between Attestation,	2

<table>
<tr><td></td><td></td><td></td><td>
<table>
<tr><td></td><td></td><td>Authentication and Identi cation, Attestation with software RoT and hardware RoT</td><td></td></tr>
<tr><td>15</td><td>Anonymous Authetication</td><td>Applicability in IoT Framework, Intel EPID Technology, PUF based Anonymity</td><td>3</td></tr>
</table>
</td></tr>
<tr><td>CS667A</td><td>Introduction To Internet Of Things And Its Industrial Applications</td><td>3-0-0-0 (9)</td><td>
Prerequisites
Knowledge equivalent of CS253 and CS455

Who Can Take The Course
PhD, Masters, 3rd and 4th year UG Students

Departments That May Be Interested
CSE, EE

Course Objective

This course will be focused on introducing students to new trends, applications, system architecture and challenges involved in developing/deploying internet of things systems using real industrial use cases. A number of systems are getting connected to the internet, where the sensor data is analyzed to monitor and control the systems. Correctly analyzing data coming from multiple sensors, choosing the right hardware given the power and performance tradeoff, hardware heterogeneity and security are some of the challenges involved in developing IoT applications. In this course, students will read research papers from top-tier conferences and journals for IoT to learn the most recent advancements in IoT research. The course will cover the real-world use cases of IoT applications and hands-on projects related to those based on the concepts learned in the class.

Course Contents

Here is a broad list of topics to be covered in the course:

<table>
<tr><th>SNo</th><th>Broad Title</th><th>Topics</th></tr>
<tr><td>1</td><td>Overview of IoT systems</td><td>New trends, applications and challenges</td></tr>
<tr><td>2</td><td>IoT system architecture</td><td>Edge devices, sensors, actuators, gateway, data storage and historical analysis in the cloud</td></tr>
</table>
</td></tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td></td>
<td>
<table>
<tr><td>3</td><td>Sensor networks</td><td>Wireless sensor networks (WSN), localization, node mobility, energy efficiency in WSN</td></tr>
<tr><td>4</td><td>Communication</td><td>MQTT, wifi, Bluetooth, RFID, LoRa, LoRaWAN, communication security</td></tr>
<tr><td>5</td><td>IoT system optimization</td><td>Low power devices, energy harvesting, performance trade-off, choosing the right hardware</td></tr>
<tr><td>6</td><td>Applications of deep learning in IoT</td><td>Introduction to deep learning, camera-based IoT applications in healthcare, retail and agriculture</td></tr>
<tr><td>7</td><td>Smart and connected devices</td><td>Raspberry-pi, Google home mini, Alexa, Echo show Industrial use cases: smart home and agriculture</td></tr>
<tr><td>8</td><td>Case studies</td><td>Smart cities, transportation, manufacturing, automobile</td></tr>
</table>
</td>
</tr>
<tr>
<td>CS668A</td>
<td>Practical Cyber Security For Cyber Practitioners</td>
<td>3-0-0-0- [9]</td>
<td>
<table>
<tr><th>Module</th><th>Topic</th><th>No. of 1hour Lectures</th></tr>
<tr><td>Introduction</td><td>Cyber Security Operations (CyOps), and integration of cyber security operations to software development and operation process (DevSecOps), and integration of relevant operations (MLops, AIops, DevOps)</td><td>5</td></tr>
<tr><td>Cyber Operation Planning and Analysis</td><td>Planning of a Cyber Security operation in an IT organization vs an OT/ICS Planning and Risk Analysis</td><td>8</td></tr>
<tr><td>Incident Detection and Characterization</td><td>Incident Indicators and Incident Detection Analyzing Incidents</td><td>4</td></tr>
<tr><td>Vulnerability and Consequence Analysis</td><td>Vulnerability Detection Methods and Tools, System Model and Consequence analysis, Threat Intelligence and</td><td>6</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr><td></td><td></td><td></td><td>
<table>
<tr><td></td><td>Threat activation of Vulnerabilities</td><td></td></tr>
<tr><td>Incident Response and Recovery</td><td>Data Analytics support for Incident Response, Backup and Recovery, Recovery from Incident Methods</td><td>4</td></tr>
<tr><td>Cloud and API security issues</td><td>Cloud Security, Cloud and API security</td><td>5</td></tr>
<tr><td>Case Studies</td><td>Industrial Case Students</td><td>8</td></tr>
<tr><td>Total Lecture hours</td><td></td><td>40 hours</td></tr>
</table>
</td></tr>
<tr><td>CS669</td><td>Design For Security</td><td>3-0-0-9</td><td>Module 1: Finite Field Arithmetic Hardware
Introduction to finite field
Finite field operations: Addition, Multiplication and Inversion
Application of finite field in cryptography
Finite field addition architectures
Finite field multiplication architectures
Finite field inversion architectures
Introduction to FPGA architecture
Module 2: Secure Constructions of Block Cipher: AES
AES S-Box Construction
Iterative AES Architecture
Power based side channel attack on AES
Side channel evaluation methodology
Side channel countermeasure: Threshold Implementation
Fault attack on AES: DFA and Trojan based
Fault attack countermeasure: Redundancy and Infection
Module 3: Secure Constructions of Stream Cipher: Grain
Introduction to linear feedback shift register (LFSR)
Berlekamp–Massey algorithm
Issues with LFSR based encryptions
Implementations of stream cipher Grain
Module 4: Efficient Implementation of Elliptic Curve Architectures
Introduction to public key cryptosystem
Introduction to elliptic curve cryptography
Elliptic curves in GF(2m)
Elliptic curves in GF(p)
Secure scalar multiplication algorithm
Montgomery elliptic curve: Implementation of Curve25519
Side channel attack on elliptic curve and countermeasure
Elliptic curve isogeny based cryptosystem for post quantum computing</td></tr>
</table>

CS671	Introduction To Natural Language Processing		A computational framework for natural language. A framework such as LFG, GPSG or Panlni in some depth. Partial description of English or an Indian language in the frame work, lexicon, algorithms and data structures for implementation of the framework. Introduction to semantics and knowledge representation. Some applications like machine translation, database interface.
CS672	Natural Language Processing Semantics		Introduction to semantics, semantic interpretation, knowledge representation, context and world knowledge, plans and actions, discourse structure, belief models, speech acts. Selected applications.
CS673	Machine Translation		Overview of Natural Language Processing; Syntax, semantics, context and world of knowledge; Strategies for machine translation, Direct, Transfer and Interlingua approaches; Rule based, Example based on Hybrid Methodologies; Construction of lexical data-base, Text generation, Machine-aided translation, user interfaces; Examples of English-Hindi and Hindi-English machine translation.
CS674	Knowledge Discovery		This course will explore different machine learning, knowledge discovery and data mining approaches and techniques: Concept Learning, Decision Tree Learning, Clustering and instance based learning, Rule induction and inductive learning, Bayesian networks and causality, Neural networks, Genetic algorithms, Reinforcement learning, Analytical learning.
CS674A	Post Quantum Security	**3-0-0-0 (9)**	• **Module 1: Quantum Computing** – Basics of Quantum Computing – Shor's Algorithm – Finite Field Operations – Karatsuba and Number Theoretic Transformation Based Multiplication – Montgomery Multiplication • **Module 2: Lattice Based Cryptography** – Basics of lattice based cryptpgraphy – NewHope, Kyber and Saber (NIST post-quantum candidates) – NTRU, NTRU Prime – Digital Signature Algorithm: Dilithium, Falcon • **Module 3: Code Based Cryptography** – Classic McEliece – HQC • **Module 4: Isogeny Based Cryptography**

<table>
<tr><td></td><td></td><td></td><td>– Supersigular Isogeny based Key Exchange (SIKE)
– Digital Signature Algorithm based on Isogeny</td></tr>
<tr><td>CS676</td><td>Computer Vision And Image Processing</td><td></td><td>Human and Computer Vision.
Image Representation and Modelling.
Line and Edge detection, labeling, Image Segmentation.
Pattern Recognition: Statistical, Structural, Neural and Hybrid Techniques.
Training and Classification.
Document Analysis and Optical Charecter Recognition.
Object Recognition.
Scene Matching and Analysis, Robotic Vision.
Role of Knowledge.</td></tr>
<tr><td>CS677A</td><td>Topics In Large Data Analysis And Visualization</td><td>3-0-0-0 (9)</td><td>
<table>
<tr><th>S. No.</th><th>Topic</th><th>Lecture hours</th></tr>
<tr><td>1</td><td>Introduction to scientific simulations and data analysis</td><td>2</td></tr>
<tr><td>2</td><td>Introduction to parallel computing</td><td>1</td></tr>
<tr><td>3</td><td>Introduction to scientific visualization</td><td>1</td></tr>
<tr><td>4</td><td>Research paper discussion on topics related to:
1. Visualization of time-varying data
2. Visualization of multivariate data
3. Flow visualization
4. Ensemble data visualization</td><td>6</td></tr>
<tr><td>5</td><td>Research paper discussion on topics related to:
5.Statistical data analysis and visualization
6.Information theory-based data analysis and visualization
7.Visualization using AI</td><td>5</td></tr>
<tr><td>6</td><td>Research paper discussion on topics related to:
8.High performance visualization
9.Performance data visualization
10.Scalable analysis and visualization</td><td>4</td></tr>
<tr><td>7</td><td>Research paper discussion on topics related to:
11.Remote visualization
12.In situ analysis and visualization
13.Resource-constrained analysis and visualization</td><td>4</td></tr>
<tr><td>8</td><td>Research paper discussion on topics related to:</td><td>3</td></tr>
</table>
</td></tr>
</table>

			14.Information visualization 15.Application-specific visualization (e.g. climate, cosmology)
CS678	Learning With Kernels		Kernel based methods in machine learning have become a major paradigm in machine learning in the last decade. The methods have also found widespread application in pattern classification problems. This course aims to first discuss the basic principles of kernel based learning methods and then branch off into some areas of current research like: techniques for finding optimal kernels, error bound analysis, novelty detection etc. Topics: Mathematical preliminaries: a) Probability - probability measures, densities, distributions, mean, variance, co-variance, sampling, stochastic process b) Linear algebra Â– vector spaces, linear combinations, convex combinations, norms, inner products, basis, inequalities c) Functional analysis Â– function spaces, norm, Banach and Hilbert spaces, basis, completeness, inequalities, reproducible kernel Hilbert spaces. Data representation, similarity, classification methods, function estimation, measures of classification performance. Kernels, representing similarity and dissimilarity. Risk and loss functions, estimators. Regularization, representer theorem. VC dimension and VC bounds. SVM and support vectors, multi-class classification, semi definite programming. Applications to biology and text categorization. Principal component analysis. Leave-1-out, leave-m-out bounds. Kernel design, hyper-kernels, optimality of kernels. Novelty detection.
CS680	Category Theory And Applications In Computing		Introduction: Basic definition and diagram; sources and sinks; monicity and epicity; isomorphisms of objects and morphisms; duality; Universal Structures: Initial terminal and zero; Category of sources and sinks; product; equalizer; regular epicity and monicity; pullback; completeness; kernel; Normal Categories: Normal hierarchy; extension of categories; factorization; chains and exactness; Morphism algebra: Biproduct; semiadditive category; Additive category; Functors: Natural transformation; categories on natural transformation; property preserving and reflecting functors; Diagram isomorphism; Similar categories; generalization of limit and colimit; H-reflection morphism and adjoint functor; Representable functors Category in context of another category;

			Application to Logic (Topoi); application to Programming Languages.
CS681	Computational Algebra And Number Theory		Elementary operations: the complexity of basic operations like additions, multiplications for integers and polynomials. Polynomials: The complexity of factorization, irreducibility testing, ideal membership etc for polynomials over finite fields. Motivating example: Reed-soloman codes. Integer Lattices: the complexity of finding a short vector in an integer lattice. Motivating example; polynomial factorization. Integers: The complexity of factorization, primality testing, discrete log computation etc for integers. Motivating examples: RSA and El Gamal cryptosystems. Elliptic curves: the complexity of addition, point counting etc. for elliptic curves. Motivating examples: Elliptic curve cryptosystems and integer factoring.
CS682	Quantum Computing		Foundations: Hilbert spaces (finite dimensional). Axioms of quantum probability. Quantum vs Classical probability. Quantum Computing: Turing machines, Boolean circuits, Quantum Circuits, Universality. Simon's problem, Phase finding, Shor's algorithm, Grovers algorithm, Probability amplification. Some applications. Quantum Information Processing: Quantum error correction.Knill-Laflamme theorem, Stabiliser codes Additional Topics: To be decided as the course progresses and if time permits.
CS684	Introduction To Algorithms And Logics In Game Theory		Games, payoffs and strategies; Two player zero sumgames, minmax theorem and computing mixed strategies via LP; Thegeneral case: Nash equilibrium: existence, algorithms to find equilibrium and complexity Issues; Examples and some other kinds of equilibriums. Social choice theory, Arrow's theorem; Elements of Mechanism design, VCG mechanism. Some extensions such as games modelswith partial information, randomized mechanisms. Extensive games: turn based games, deterministic and stochastic games, winning conditions, determinacy, solving these games. Logics to reason about games and strategies: Alternating time temporal logic and its extensions.
CS685	Data Mining		We are witnessing an unprecedented growth in the amount of data, starting from protein sequences and structures to biomedical images, sensor readings and chemical data. In order to render this vast

			amount of data more useful than just a digital data storage structure, the ability to mine for knowledge inherent in the collection must be supported. This course will cover the standard algorithms for such data mining techniques. Special emphasis will be given on the recent trends in mining text data, mining graphs, mining spatio-temporal data, etc. Besides the lectures by the instructor, the students will be asked to coordinate a dis-cussion about a recent paper in the class. They will also be required to complete a group project that will provide them with a hands-on experience on working with the techniques taught in the class.
CS686	Data Driven Program Analysis		The course will cover recent applications of data mining techniques such as frequent itemset mining, anomaly detection, and classification in analyzing programs for automated testing, debugging, fault isolation, repair, synthesis, etc. The detailed contents are: Overview of interesting problems in programming languages and software engineering; motivation of applying data-driven techniques for addressing these problems. Fundamentals of data mining: Data pre-processing, statistical measures, classification and clustering, frequent itemset mining, anomaly detection, graph mining. Fundamentals of program analysis: Control-flow analysis, dataflow analysis, program dependence graphs, efficient profiling algorithms, static and dynamic instrumentation techniques. Data mining for program analysis: Data-driven techniques for bug localization, program synthesis and repair, invariant generation, automated testing and debugging.
CS687	Algorithmic Information Theory		I'll provide you with a bibliography as the course progresses. 1. Introduction Motivation and History, Brief Review of Prerequisites 2. Kolmogorov Complexity Plain Kolmogorov Complexity and its Problems, Self-delimiting machines and Kolmogorov Complexity, Symmetry of Information, the Coding Theorem, Algorithmic Probability. 3. Randomness. Uniform Distribution, Computable Distributions, Definition of a finite random string, Abundance of randomness, Dentin of an finite random sequence, Martingale Characterizations. 4. Information Theory Information Theoretic Inequalities, Algorithmic Entropy, Randomness and Complexity.

			5. The Incompressibility method Some basic lower bounds, The incompressibility method, A lower bound for Shell sort. 6. Resource-bounded Kolmogorov Complexity Hartmanis' results, Levin's Optimal Search, A Hierarchy theorem for Probabilistic Classes with advice, Hutter's Optimal Search, Kolmogorov Complexity and Communication Complexity.
CS688	Computational Arithmetic-Geometry And Applications		We want to count the number of roots of an algebraic system (over finite fields). This is a very difficult question in general (eg. #P-hard). However, there are fast algorithms known for special cases. In this course we will focus on the "2-variable" case, i.e. curves. This case already demands significant theory and has an amazing list of applications in computer science. We will cover some important aspects of the theory in a self-contained way, and see as many applications as time permits. . Fundamentals: - Algebraic-geometry notation - Zeta function of curves over finite fields - The relevant Riemann hypothesis - Finally, its proof and Weil bound for curves Applications: - Counting points on curves - Integer factoring via curves - Hyperelliptic curve cryptography - Algebraic geometry codes - Computing roots of unity in finite fields
CS689A	Computational Linguistics For Indian Languages	**3-0-0-0 (9)**	Natural language understanding, processing and conversation have always been a goal for computer science. This course will cover the basics of how a computer deals with this subject. The course will start from the basic units of a language (words or tokens). It will also include more semantic order tasks such as lemmatization, parts-of-speech tagging, co-reference resolution, parsing a sentence, etc. Higher order tasks such as question-answering, discourse analysis, etc. will be covered as well. The field of natural language processing has made enormous advancements mostly due to deep learning algorithms. However, availability of large amounts of data along with annotations is a necessity for deep learning. In this course, the focus will be on Indian languages, where abundance of such high quality data is still not very common. In addition, traditional theories of what a word or a sentence constitutes and how a sentence should be parsed/understood will be covered.
CS690	Computational Genomics		**Computational genomics** is a novel and very active application field of computer science where

			biological mechanisms are deciphered from **genome sequencing data** using computational and statistical analyses. In the past twenty years, an explosion of genomic data (from human and several other organisms) has revolutionized a number of subfields of biology – cell and molecular biology, developmental biology, disease biology and so on. Computer science plays a central role in genomics – from sequencing and assembling of DNA and RNA sequences to analyzing genomes (or transcriptomes) for elucidating diverse biological mechanisms through innovations in machine learning, data structure and algorithms. In this course, you will be introduced to some of the most seminal machine learning and algorithmic approaches for sequence analysis as well as the most recent advances in the field. The course will be structured as a combination of lectures and discussion of recent publications in the field. The lectures will introduce the topics and seminal algorithms followed by research paper discussions on advanced and most recent developments. Tentative Topics **Fundamentals of Biological Sequence Analysis:** Sequence alignment algorithms, Hidden Markov Models (HMMs) and modeling of biological sequences **Genome-Scale Index Structures:** Suffix array, Suffix tree, Burrows-Wheeler transform (BWT), BWT index and applications **Genome-Scale Short Read Alignment:** Dynamic programming along suffix tree paths, short read alignment algorithms **Genome Assembly Algorithms:** De Bruijn graphs **Variant Calling Algorithms:** Probabilistic dynamic programming **Transcriptomics:** Gene expression analysis, normalization, differential expression analysis, clustering **Single-cell omics:** Dimension reduction algorithms, Generative processes, Manifold learning, Trajectory inference, Regulatory networks **Cancer Genomics:** Probabilistic graphical models for tumor heterogeneity analysis, somatic mutation detection algorithms, driver mutation detection, matrix factorization problems in cancer genomics **Deep Learning in Genomics:** ChIP–seq data, **transcription factor binding sites,** Graph-convolutional neural networks, transfer learning **Phylogenetics:** Markov models of molecular evolution, character-based phylogeny algorithms –

			maximum parsimony, maximum likelihood and Bayesian inference
CS697	Special Topics In Computer Science & Engineering		Special and advanced topics in different areas of Computer Science and Engineering will be covered under this course.
CS698A	Selected Areas Of Mechanism Design	3-0-0-0 (9)	This course is based on selected topics in mechanism design. These topics include stable matching, auctions, selfish routing, games on networks, potential games. This is a research-oriented course, hence students are expected to read and present cutting-edge research topics in this area, and also develop writing skills towards a formal technical report. The tentative plan of coverage is as follows. Part 1: Mechanism design theory – quick recap Stable matching theory Auctions Voting and fair division – cake cutting Algorithmic aspects of Mechanism Design Network Games Cooperative Games Some topics from students' interest Part 2: Selected papers from leading conferences and journals on the topic that deals with research in mechanism design in the paradigm of artificial intelligence and multi-agent systems Part 1 is the lecture part by the instructor. Part 2 is for the students to survey, read, present papers (with help from the teaching staff) on the state-of-the-art of the topics that includes (but are not limited to) those discussed in Part 1. This part also requires every student to do a course project (aim: to extend the state-of-the-art, deliverables: writing a technical report comparable to a formal publication and an end semester presentation). Evaluation: One exam (midterm) at the end of part 1. For part 2, the technical report and the quality of the presentations will be counted for the evaluation. Both parts have equal weightage.
CS698B	Linear Algebraic Tools For TCS	3-0-0-0-9	The course will focus on two linear algebraic tools, convex optimization and spectral graph theory, which have played an important role in the development of theoretical computer science. There will be broadly three parts of the course. The first few weeks will be devoted to the basics of linear algebra. It will mostly be focussed on rank, eigenvalues and eigenvectors of matrices. This will prepare students for the two tools they will be studying later. The first tool will be convex optimization and its applications in the field of complexity theory. Initially, we will look at linear

			programming. Through linear programming, we will introduce the concepts of convexity, optimization and duality. We will move to semidefinite programming later and generalize the concepts studied before. Finally, we will show some applications of linear as well as semidefinite programming in complexity theory. This part will introduce them to the concepts of relaxation and rounding. The second tool we will talk about is spectral graph theory. We will study graphs by looking at the spectrum of the associated matrices. It is surprising that many of the combinatorial properties of the graph can be studied by looking at the eigenvalues and eigenvectors of the Laplacian matrix of the graph. We will also discuss spectral sparsification, which helps in designing solvers for linear equations.
CS698C	Topics In CSE: Sketching And Sampling For Big Data Analysis		Today, there is a tremendous interest in processing “Big Data” due to the ubiquity of large data sets being generated in production systems. This has led to an interest in designing algorithms and systems for problems with very large inputs. The problems range from matrix problems and numerical linear algebra, optimization problems and graph problems. This course will highlight the recent advances in algorithms for numerical linear algebra and optimization that have come from the technique of linear sketching, whereby given a matrix, one first compresses it to a much smaller matrix by pre-multiplying it by a (usually) random matrix with certain properties. Much of the expensive computation can then be performed on the smaller matrix, thereby accelerating the solution for the original problem. This technique has led to the fastest known algorithms for fundamental problems in this area. We will consider least squares as well as l1-regression problems, low rank approximation, and many variants of these problems, such as those in distributed environments. We will also discuss connections of these methods with and using graph sparsifiers. Some of this work is partially covered in the monograph by Prof. David Woodruff: “Sketching as a Tool for Numerical Linear Algebra”, Foundations and Trends in Theoretical Computer Science, vol 10, issue 1-2, pp. 1-157, 2014, publisher: NOW Publishing. Prof. David Woodruff will be giving a GIAN course on this topic from Jan 2-6. Each day will have between 3-4 hours of lecture, except perhaps the final day. All students who wish to take the course will be enormously benefitted by attending Prof. Woodruff's lectures. It is mandatory for those registering in the course to also attend the GIAN lectures.

CS698D	Topics In Data Compression		This course covers the introductory basics and some recent topics in the area of lossless data compression. We will cover the following topics. First, we introduce the notion of a lossless compressor. We will cover the fundamental techniques like Shannon-Fano coding, Huffman coding and Arithmetic co des. We will cover Kolmogorov complexity as the "limit notion" of a lossless code, where the compressor can be any arbitrary program. Special emphasis will be given to the class of "universal codes", including the Lempel-Ziv algorithm and the Burrows-Wheeler transform. We then study the optimality and rate of convergence results of the Lempel-Ziv algorithm, in the weak sense, and in the strong sense. We then cover some current topics in the topic of universal codes. Specifically,the class of universal codes has been shown to be "non-rob ust" with respect to small perturbations. We will study the tools used to establish these results, and investigate some open problems in this area. Finally,we finish with some applications of the Lempel-Ziv a lgorithm to areas like Statistical inference, and algorithms. Mathematical techniques used in these results i nclude recurrence theorems from ergodic theory, and "Rokhlin-Kakutani tower constructions" from Ergodic theory (also known as "cutting and stacking". If time permits, we will consider the recent class of optimal compressors called "Asymmetric Numeral Systems", which were introduced in 2014.
CS698E	Topics In Computer Architecture And Operating Systems		Selected research papers from recent flagship conferences such as ISCA, ASPLOS, MICRO, OSDI, and SOSP. Each student has to review all papers before the in-class paper discussion. One student would act as the lead (for one paper) for the discussion and will kick-start the discussion with a presentation. Papers will be made available before the first class of the course. Assessment Policies 50 = Research Project 25 = Paper review 20 = Paper presentation (leading the discussion) 5 = Classroom participation
CS698H	Topics In Homotopy Type Theory		The course will assume some background in HoTT, (Ch 1, 2, 5, 6.1-6.5 from HoTT book). We plan to cover in the course some of the remaining material including applications in the HoTT book.

			We also plan to cover some issues in semantics of HoTT and more recent topics like cubical type theory. Students are expected to participate proactively in the course by discussion and by offering to give some talks.
CS698I	Relational Structures In Games	3-0-0-0 (9)	Game theory, a field which originated in the intersection of mathematics and economics offers models to analyse strategic interaction of rational agents. Game theoretic models are often used as tools in the analysis of multi-agent systems in computer science. From the perspective of computer science and artificial intelligence, representation and computational complexity of various game models is a pertinent issue. Representing multiplayer games using many of the standard game theoretic models studied in economics (like strategic games) is problematic for various reasons. The parameters needed to represent games in such models usually grows exponentially with the number of players. Such game models also abstract the underlying structure present in the multiplayer game, which typically plays a crucial role in the algorithmic analysis. Thus representations which are compact and those that naturally model the underlying structure of multiplayer games are important. The main goal of this course is to study the representational and algorithmic aspects of models of multiplayer interaction. This includes game theoretic models as well as those studied in social choice theory. We will primarily study topics and address questions at the interface of theoretical computer science and economics. The topics include the following. 1. Basics of game theory - models and solution concepts 1. Strategic form games, pure and mixed strategies 2. Extensive games with perfect information 3. Dominance and equilibrium 2. Models with compact representations 1. Graphical games 2. Polymatrix games 3. Coalition formation games 4. Coordination games 3. Algorithmic analysis 1. Hardness in computation of equilibria 2. Efficient computation via structural restrictions 3. Algorithmic analysis of various fair division models

<table>
<tr>
<td>CS698J</td>
<td>Introduction To Constructive Types Theory And Its Applications</td>
<td>3-0-0-9</td>
<td>We plan to cover the following topics.
1. Natural deduction for propositional logic, Curry Howard isomorphism.
2. Simply Typed lambda calculus, its extensions with polymorphism and dependent types. Normalization theorems.
3. Some issues in semantics, an introduction to category theory to the extent needed in semantical issues.
4. Intuitionistic Type theory of Martin Lof. Proof relevance, extensional vs intensional, identity types. Path induction. Groupoid model for intensional type theory.
5. An introduction to homotopy type theory. Type equivalence and univalence. higher inductive types. Some recent developments like computational type theory will be explored if time permits.

The course will be theoretical. It will encourage active involvement of students in terms of reading papers, students presentations and discussions. These will also be main part of evaluation.</td>
</tr>
<tr>
<td>CS698K</td>
<td>Designing Verifiably Secure Systems</td>
<td>3-0-0-0-9</td>
<td>
<table>
<tr><th>S No</th><th>Broad Title</th><th>Topics</th><th>Num. Lec.</th></tr>
<tr><td>1.</td><td>Satisfiability</td><td>propositional logic, DPLL/CDCL/Chaff, SAT Extensions: QBF, AllSAT, etc.</td><td>3</td></tr>
<tr><td>2.</td><td>SMT and applica-tions</td><td>first-order logic, SMT, approaches to SMT solving, symbolic execution using SMT</td><td>4</td></tr>
<tr><td>3.</td><td>Cryptograp hic Primitives</td><td>message authentication, symmetric-key encryption, public-key cryptography, modeling crypto in SMT</td><td>3</td></tr>
<tr><td>4.</td><td>Model Checking</td><td>explicit state, symbolic bounded reachability, induction, abstraction/refinem ent, CEGAR, self-composition and hyper- properties, simulation/bi-simulation, applicati ons to security verification</td><td>5</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr><td></td><td></td><td></td><td>
<table>
<tr><td>5.</td><td>System Security Primitives</td><td>isolation, access control, principle of least privilege, implementations of these in HW and SW, verification of these primitives: noninterference, static and dynamic information flow tracking</td><td>4</td></tr>
<tr><td>6.</td><td>Case studies, synthe-sis, machine learning</td><td>case studies with potential examples being enclave platforms, hypervisor security, e-voting; syntax-guided synthesis, applica- tions to/of machine learning and statistical inference</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>CS698N</td><td>Recent Advances In Computer Vision</td><td></td><td>In this course, we look at a subset of topics in the following exciting sub-areas of research in Computer Vision. This list of topic is adaptable depending on the level and interests of the students actually taking the course.
Human Analysis eg. actions, pose estimation, facial analysis, attribute recognition, pedestrian detection
Language and Vision eg. image captioning, visual question answering
Image segmentation eg. semantic segmentation and multi resolution edge estimation, instance segmentation
There will be a significant project component -- the students are expected to mainly learn by doing.</td></tr>
<tr><td>CS698O</td><td>Special Topics In Natural Language Processing</td><td>3-0-0-0-9</td><td>Pre-Requisites:

Instructor's consent and Must: Introduction to Machine Learning (CS771), Proficiency in Linear Algebra, Probability and Statistics, Proficiency in Python Programming

Desirable:

Probabilistic Machine Learning (CS772), Topics in Probabilistic Modeling and Inference (CS775), Deep Learning for Computer Vision (CS776)

Departments Which May Be Interested:

CSE, EE, MTH, IME, ECO</td></tr>
</table>

			Level Of The Course: Senior UG and PG (6xx level) Course Description: Natural language (NL) refers to the language spoken/written by humans. NL is the primary mode of commu- nication for humans. With the growth of the world wide web, data in the form of textual natural language has grown exponentially. This calls for development of algorithms and techniques for processing natural language for the purposes of automation and for the development of intelligent machines. This course will primarily focus on understanding and developing techniques/learning algorithms/models for processing text. We will have a statistical approach to Natural Language Processing (NLP), wherein we will learn how one could develop natural language understanding models from regularities in large corpora of natural language texts. Tentative Topics: 1. Introduction to Natural Language (NL): why is it hard to process NL, linguistics fundamentals, etc 2. Language Models: n-grams, smoothing, class-based, brown clustering 3. Sequence Labeling: HMM, MaxEnt, CRFs, related applications of these models e.g. Part of Speech tagging, etc. 4. Parsing: CFG, Lexicalized CFG, PCFGs, Dependency parsing 5. Applications: Named Entity Recognition, Coreference Resolution, text classification, toolkits e.g. Spacy, etc. 6. Distributional Semantics: distributional hypothesis, vector space models, etc. 7. Distributed Representations: Neural Networks (NN), Backpropogation, Softmax, Hierarchical Softmax 8. Word Vectors: Feedforward NN, Word2Vec, GloVE, Contextualization (ELMo etc.), Sub-word information (FastText, etc.) 9. Deep Models: RNNs, LSTMs, Attention, CNNs, applications in language, etc. 10. Sequence to Sequence models: machine translation and other applications 11. Transformers: BERT, transfer learning and applications

CS698P	Applications Of Markov Chains In Combinatorial Optimization And In Evolutionary Dynamics	3-0-0-0-9	The course deals with applications of Markov chains techniques in certain areas of computer science and of biology. The unifying aspect in these applications is the role played by mixing time analysis, which is the focus of the course. In a typical combinatorial optimization problem, each instance x is associated with a state space Sx, usually of size exponential in the size of x, where each element of Sx has an associated cost (or a value). The problem is to nd a state with minimum cost (or with maximum value). In the Markov chains approach to solving such a problem, a Markov chain is associated with each instance with the property that the goal state has the highest probability in the stationary distribution of the chain. Success of this approach crucially depends on the mixing time of the associated chain, that is, how quickly does the chain come close to its stationary distribution. Markov chains have also been used to model evolution for the nite population case lending themseves to stochastic e ects. For a population of size N of m types, a state of the chain is the labelling of the N individuals each into one of the m types. The population goes from one state to another through reproduction, mutation, and selection. The speci c way in which each of these happens determine the the transition probability matrix of the chain. The result of the evolution modelled by the chain is given by the stationary distribution of the chain. In most cases however, it is not known how to determine the stationary distribution directly, we need to simulate the chain `suciently long' and then sample from the resulting distribution to obtain statistical properties of the distribution. We need to determine the mixing time of the chain to nd out how long is `suciently long'. It has been established recently that the expected motion of such evolutionary chains turns out to be a dynamical system, the trajectory of which determines the mixing time properties like rapid mixing. Approximate outline of the course: Ergodicity theorem of Markov chains. Mixing time of Markov chains. Combinatorial optimization using Markov chains. Metropolis algorithm. Notion of conductance and its relation to rapid mixing. Necessary and sucient conditions for Markov chains approach to succeed for combinatorial optimization. Markov chains modelling of molecular evolution. Quasispecies

			model. Viral evolution and notion of error threshold. Evolution for the nite population case. Expected motion of evolutionary Markov chains and corresponding dynamical system. Detailed analysis when the population is of two types. A qualitative understanding of the general m types case.
CS698Q	Complexity Measures For Boolean Functions	**3-0-0-0 (9)**	Prerequisites: **Linear algebra.** Outline The analysis of Boolean functions has been an important tool in many diverse areas of computer science, e.g., complexity theory, property testing, social choice theory and quantum computing. The aim of this course is to provide an introduction to the analysis of Boolean functions, different complexity measures on these Boolean functions and the relationship between these complexity measures. We will start with a Fourier analysis on the Boolean hypercube, how it is defined and what are its important properties. These properties will be accompanied by applications in different areas of computer science. We will cover the concept of Fourier degree, sign degree and approximate degree too. This background material and applications will constitute around half of the course. The next half will start with combinatorial complexity measures on Boolean functions like sensitivity, block sensitivity and certificate complexity. We will then talk about complexity measures coming from the computational world like decision tree complexity, randomized decision tree and quantum query complexity. The final part of the course will deal with relations known between these complexity measures from different domains. Most of these were polynomially related and we will give proof of these relationships. If time permits, we will cover the recent breakthrough result of Huang which shows that even sensitivity is polynomially related to other complexity measures.
CS698R	Deep Reinforcement Learning	3-0-0-0 (9)	Pre-Requisites Instructor's consent and **Must:** Solid understanding of Machine Learning (e.g, CS771 or equivalent course), theoretical and practical knowledge of Deep Learning, Proficiency in Linear Algebra, Probability and Statistics, Proficiency in Python Programming. Desirable

			Probabilistic Machine Learning (CS772), Topics in Probabilistic Modeling and Inference (CS775), Deep Learning for Computer Vision (CS776) Level Of The Course Senior UG and PG (6xx level) Course Description In this course, we will explore how an agent (via interactions with the environment) can learn by trial and error. This is quite different from supervised machine learning and comes close to how humans learn by interactions. Reinforcement Learning (RL) deals with problems that require sequential decision making. This course will explore the foundations of reinforcement learning. We will study different algorithms for RL and later in the course, we will explore how functional approximation in RL algorithms could be done using neural networks giving rise to deep reinforcement learning. Tentative Topics Introduction to Reinforcement Learning Multi-armed bandits Markov Decision Processes Dynamic Programming Monte Carlo Methods Temporal-Difference Learning Function Approximation Methods Policy-Based and Value-Based Algorithms: REINFORCE, SARSA, DQN, Advantage Actor-Critic (A2C) Proximal Policy Optimization (PPO)
CS698V	Introduction To Lambda Calculus, Types And Models		Lambda calculus was proposed by Church as a syntactic system for manipulating constructive notion of functions. Its simplicity and expressive power is remarkable. It also gives rise to questions such as self application and non-termination of reductions. Types are used to annotate terms with more tractable properties. Many type systems, such as simple types, intersection types and polymorphic types have been studied these also relate to types in programming languages. Extensions of lambda calculs are often used to give operational semantics of theoretical core of various programming languages. Semanics (or model) of lambda calculus, using traditional sets was challenging as it requires embedding a suitable function space over a set D witin D so that function abstraction and applications are definable. Several models were proposed, finally a satisfactory model D1 was found, as a limit of

			some finite constructions, whose elements have computational significance. This led to development of domain theory, which is also used as a foundation for a theory of computation and semantics of programming languages. In the course, we will look at these ideas. The course will be theoretical and plans to follow the following text closely.
CS698W	Topics In Game Theory And Collective Choice		This course deals with topics at the interface of Economics and Computer Science. The focus will be more on the applications of game theory in social decision making. For example, how online advertising slots are allocated among competing advertisers or how the mobile telephony spectrum is distributed among the competing service providers such that certain “good” and “fair” properties are satisfied. Problems of similar flavor exist in many more applications like crowdsourcing, internet routing, fair division of goods, matching of students to advisors, facility location, social networks and many more. To understand these applications and to improve them, technology needs to partner with economic principles that drive them. This course is aimed to develop those economic principles. Even though the course is mainly focused on mechanism design (inverse game theory), it does not assume any background on game theory. The basic concepts will be developed in the initial phase of the course. The later part will see a bit of cooperative game theory and several application domains of these ideas. There are no specialized prerequisites for this class. I will assume familiarity with formal mathematical reasoning, some probability theory, basic calculus, the basics of computational complexity. I believe that one learns the concepts of an algorithm better when one codes that algorithm. Therefore, experience in programming will be useful. Detailed plan of the course: PDF Here is the course timetable (for figuring out clashes with other courses etc.) Announcement(s): Lecture-Scribe assignment is now available. Scribes will be updated at the beginning of the week. Some projects ideas are shared. Check Piazza. Problem set 1 is available (few additions/edits may be made later). Don’t send us solutions, these are for your practice only. **Midterm exam is on September 22, 2017, from 1-3 PM in KD 101. It will be a closed book/notes exam.** **Midterm**: questions, selected solutions. **No class on Oct 27, 2017** — Antaragni Friday! Course logistics:

			Class times and venue: Tue Fri 12.00 – 13.00, Wed 14.00 – 15.00, KD 102 **Instructor:** Swaprava Nath (office hours: by appointment, mail at swaprava@cse.iitk.ac.in with [CS698W] in the subject) **TA:** Rahul Jain (office hours: email at jain@cse.iitk.ac.in, Piazza is better to communicate though) **Evaluation:** 1 Midterm exam, 1 Final exam (both closed book), 1 group course-project (30% each) + scribing around 2 lectures (will follow roughly the procedure mentioned here) — 10%. **Reference texts:** No specific one. The following two books could be helpful. **"Game Theory and Mechanism Design"** — Y. Narahari, World Scientific and IISc Press, paperback. **"**Multiagent Systems**"** — Y. Shoham and K. Leyton Brown, Cambridge University Press, **Lectures** [Will be updated as the classes go along. *Disclaimer*: these scribed lecture notes resulted from a compilation from multiple other sources, e.g., books and other lecture notes.] **Lecture 1:** Introduction to the course with examples of game theory in practice. [slides] [badminton game video] **Lecture 2:** Game theory 1: examples of games, preferences without utility representation, von-Neumann-Morgenstern utilities. [scribed notes] **Lecture 3:** Game theory 2: vNM theorem, rationality and intelligence, common knowledge, dominant strategy, ideas of equilibrium. [scribed notes] **Lecture 4:** Game theory 3: Pure and mixed strategy Nash equilibrium, best response interpretation, examples of mixed Nash in games, characterization theorem of MSNE. [scribed notes] **Lecture 5:** Game theory 4: proof of characterization theorem, examples of its use to find MSNE, computation of MSNE — difficulty, Nash theorem. [scribed notes] [MSNE hardness paper] **Lecture 6:** Game theory 5: Nash theorem and its proof. [scribed notes] **Lecture 7:** Game theory 6: correlated equilibrium, comparison with MSNE, extensive form games — notation and examples. [scribed notes] **Lecture 8:** Game theory 7: equilibrium refinement for perfect information EFG, subgame perfect Nash equilibrium (SPNE), computing SPNE — backward induction, expressive power of PIEFG — imperfect information EFG. [scribed notes] **Lecture 9:** Game theory 8: IIEFG — richer representation of games, richer strategy space, mixed versus behavioral strategies, incomparability

			of mixed and behavioral strategies — games with perfect recall. [scribed notes] **Lecture 10:** Game theory 9: outcome equivalence of mixed and behavioral strategies in games with perfect recall, equilibrium notion, "equilibrium notion tied to the belief of the players", Bayesian belief, sequential rationality, perfect Bayesian equilibrium. [scribed notes] **Lecture 11:** Game theory 10: classification of standard games, application domain: peer-to-peer file sharing. [scribed notes] [paper 1] [paper 2] [slides] **Lecture 12:** Game theory 11: games with incomplete information — Bayesian games, types, common prior, ex-ante, ex-interim utilities, examples — bargaining, auction. [scribed notes] **Lecture 13:** Game theory 12: Bayesian games — equilibrium concepts, Nash, Bayesian equilibria, equivalence, existence of Bayesian equilibrium. [scribed notes] [addendum] **Lecture 14:** Game theory 12.5: examples of Bayesian equilibrium in first and second price auctions, Mechanism Design — examples and notation. [scribed notes] **Lecture 15:** Mechanism design 1: social choice function, mechanisms and implementation via indirect and direct mechanisms, dominant strategy implementability and dominant strategy incentive compatibility, revelation principle for DSI SCFs. [scribed notes] **Lecture 16:** Mechanism design 2: implementation in Bayesian equilibrium, Bayesian incentive compatibility, revelation principle for BIC, Arrovian social welfare functions, preference ordering, weak/strong Pareto. [scribed notes] **Lecture 17:** Mechanism design 3: independence of irrelevant alternative, Arrow's impossibility theorem and its proof, decisive group, field expansion lemma. [scribed notes] **Lecture 18:** Mechanism design 4: group contraction lemma — finishing Arrow's theorem, social welfare function to social choice function, voting rules. [scribed notes] **Lecture 19:** Mechanism design 5: axioms for social choice function — Pareto efficiency, unanimity, onto-ness, monotonicity, strategyproofness, equivalence of SP and MONO, equivalence of PE, UN, and ONTO under SP. [scribed notes] **Lecture 20:** Mechanism design 6: Gibbard-Satterthwaite theorem and its proof (for two voters), cases where GS theorem does not hold. [scribed notes] [Ref: Chapter 6 of the lecture notes by Debasis Mishra] **Lecture 21:** Mechanism design 7: GS theorem and

			unrestricted domains, domain restrictions, single peaked preferences, example of non-dictatorial strategyproof SCFs, median voter SCF. [scribed notes] **Lecture 22:** Mechanism design 8: properties of SCF in single-peaked domain, monotonicity, anonymity, Moulin's characterization theorem with median voter rule. [scribed notes] **Lecture 23:** Mechanism design 9: proof of Moulin's characterization theorem with median voter rule. [scribed notes] **Lecture 24:** Mechanism design 10: private good allocation — task sharing model, Pareto efficiency, Anonymity, strategyproofness, some candidate SCFs. [scribed notes] **Lecture 25:** Mechanism design 11: uniform rule SCF, characterization of Pareto efficiency, anonymity, and strategyproofness using this rule, mechanism design with transfers and quasi-linear preferences. [hwnotes] [scribed notes (draft)] **Lecture 26:** Mechanism design 12: examples of allocation and payment rules in quasi-linear domain, dominant strategy incentive compatibility and its impact on the payment functions. [scribed notes] **Lecture 27:** Mechanism design 13: Pareto optimality of mechanisms in quasi-linear domain, relation with allocative efficiency, implementing AE rules — Groves class of payments, Clarke's pivotal rule. [scribed notes] **Lecture 28:** Mechanism design 14: illustration of VCG mechanism and its properties in single-object allocation, public good allocation, combinatorial allocation. [hwnotes] [scribed notes (draft)] **Lecture 29:** Mechanism design 15: more properties of VCG mechanism in combinatorial good allocation, case study: Internet advertisements, kinds of ads, position auctions, agent valuations, click-through-rate, allocation rules. [hwnotes] [scribed notes (draft)] **Lecture 30:** Mechanism design 16: Internet advertisements — sponsored search auctions, comparison of VCG and generalized second price (GSP) mechanisms, desirable properties of VCG. [hwnotes] [scribed notes (draft)] **Lecture 31:** Mechanism design 17: limitations of VCG, generalization of VCG — affine maximizer allocation rules, implementability, Roberts' theorem. [hwnotes] [scribed notes (draft)] **Important:** Lecture-Scribe assignment. Scribing Here is the template for scribing the lectures. Here and here are good introductions to LaTeX. Please email me the scribed lecture notes within 2 days of the class (first week scribes get one

			week's time) — I'll immediately put them on the course page as 'draft'. Later when I review the notes, you may need to update the notes and resubmit. *Less the update that is needed, better is the credit — so consider to do the first draft carefully.* Course Project Since this is a research-focused course, the course project is extremely important for developing new ideas and transforming them into workable solutions. It is seen that in doing a project, where a learner is required to either code a system or prove a result independently, s/he learns very intricate details of an idea or concept. A course project can be (a) completely a theoretical development, (b) completely a real-world application development, or (c) a mix of the previous two. All topics has to have a significant game-theoretic/mechanism design component — however there is no restriction on what the application area may be. It is a good idea to keep looking for ideas when different topics are discussed in the class — and if you have an idea that may be converted into a project, come and talk to me. Deadline for submitting the project proposals will be announced later. Virtual Classroom This semester we will be using Piazza for class discussion. The system is highly catered to getting you help fast and efficiently from classmates, the TA, and myself. Rather than emailing questions to the teaching staff, I encourage you to post your questions on Piazza. Enrolment link (students and TAs, please register yourself here) Class link
CS698X	Topics In Probabilistic Modeling And Inference		Probabilistic models for data are ubiquitous in many areas of science and engineering, and specific domains such as visual and language understanding, finance, healthcare, biology, climate informatics, etc. This course will be an advanced introduction to probabilistic models of data (often through case studies from these domains) and a deep-dive into advanced inference and optimization methods used to learn such probabilistic models. This is an advanced course and ideally suited for student who are doing research in this area or are interested in doing research in this area. Pre-Requisites Instructor's consent. The course expects students to have a strong prior background in machine learning and probabilistic machine learning (ideally through formal coursework), probability and statistics, linear algebra, and optimization. The students must also be proficient in programming in MATLAB, Python, or R.

			Topics A tentative list of topics to be covered in this course includes Fundamentals of probabilistic modeling Basics of probability distributions and their properties Basics of probabilistic inference: MLE/MAP/Bayesian inference Probabilistic graphical models (directed and undirected models) Probabilistic approaches for linear modeling, Sparse Bayesian Learning Latent variable models Mixture models and latent factor models Latent variable models for dynamic/sequential data Latent variable models for networks and relational data Latent variable models with covariates Approximate Inference Expectation Maximization MCMC methods Variational methods Scalable inference with stochastic optimization Other methods: Likelihood-free methods, spectral methods, etc. Nonparametric Bayesian methods Gaussian Process for function approximation Dirichlet process and beta processes Other stochastic processes (gamma/point processes, etc., and their applications) Bayesian Optimization Bayesian Deep Learning Theory of Bayesian statistics Probabilistic programming Other topics based on students' interests Treatment of the above topics will be via several case-studies/running-examples, which include generalized linear models, finite/infinite mixture models, finite/infinite latent factor models, matrix factorization of real/discrete/count data, sparse linear models, linear Gaussian models, linear dynamical systems and time-series models, topic models for text data, etc.
CS698Y	Modern Memory Systems		The course comprises of four main modules apart from a module on preliminaries. Module 0: Preliminaries on Processor, Cache, and Memory Module 1: Shared on-chip Cache Management (i) Cache management policies (insertion, eviction, promotion, and bypassing) (ii) Multi-core cache hierarchies (inclusive, exclusive, non-inclusive), static/dynamic non-uniform cache hierarchies

			(iii) Latency tolerance techniques (hardware prefetching and cache compression) Module 2: DRAM Systems (i) DRAM controllers, timing constraints, DRAM organization, DRAM modeling issues (ii) DRAM scheduling policies and DRAM address mapping schemes (iii) Management of DRAM capacity, bandwidth, energy, and power Module 3: Secure/reliable Memory (i) Side/covert-channel attacks, cold boot attacks at the cache and DRAM (ii) DRAM scaling challenges, reliability issues, and row-hammer problem Module 4: Emerging Topics (i) 3D/2.5D DRAM stacking, DRAM cache, PCM, 3D X-point, and processing-in-memory (ii) Memory hierarchy for approximate computing and memory for large-scale systems
CS699	MTech Thesis		
CS711	Introduction To Game Theory And Mechanism Design	3-0-0-0 (9)	This course is an introduction to classical game theoretic ideas and results with the aim to design mechanisms that satisfy desirable axioms. The course will introduce ideas of rationality and intelligence, cover non-cooperative games (including complete information simultaneous move and sequential games, and later incomplete information games), cooperative games (ideas of coalition, core, Shapley value, neucleolus etc), and introduce mechanism design framework (social choice and welfare functions, Arrow's impossibility result, unrestricted preferences and Gibbard-Satterthwaite result, domain restriction: single-peaked, task allocation domains, quasi-linear preferences), and demonstrate applications of these ideas in practice. A tentative list of topics are as follows. 1. Non-cooperative game theory 1. Quantitative models of strategic interaction: rationality, intelligence, common knowledge 2. Complete information simultaneous move games – normal form representation 1. Ideas of equilibria: domination of strategies, Nash equilibrium 2. Existence results for mixed and pure Nash equilibrium 3. Correlated equilibrium.

			3. Complete information sequential move games – extensive form representation 1. Perfect and imperfect information extensive form games 2. Equilibria concepts – subgame perfect equilibrium, perfect Bayesian equilibrium, analogies with pure and mixed Nash equilibrium 4. Incomplete information games 1. Bayesian games 2. Equilibria concepts tied to the belief system 3. Nash and Bayesian equilibria in incomplete information games 5. Cooperative Game Theory 1. Utility representation in form of coalition 2. Transferable utilities game 1. Imputation, core, Shapley value, nucleolus 6. Introduction to mechanism design 1. Incomplete information to player types 2. Social welfare function, Arrow's impossibility result 3. Social choice function, Gibbard-Satterthwaite result 4. Domain restriction 5. Single-peaked preferences 6. Task allocation domain 7. Quasi-linear preferences 7. Some real world applications of mechanism design
CS712	Selected Areas Of Mechanism Design	3-0-0-0 (9)	Pre-Requisites: Familiarity with formal mathematical reasoning, probability theory, calculus, basics of computational complexity, and computer programming. The course also expects familiarity with game theoretic ideas and results – hence a course like CS711 or CS656 will be required. Description This course is based on selected topics in mechanism design. These topics include stable matching, Internet advertising, sponsored-search auctions, selfish routing, games on networks, potential games. This is a research-oriented course, hence students are expected to read and present

			cutting-edge research topics in this area, and also develop writing skills towards a formal technical report. **The tentative plan of coverage is as follows.** **Part 1:** Mechanism design theory – quick recap Stable matching theory Quasi-linear domain Internet advertising Auctions Single-parameter domains Fair division Algorithmic aspects of mechanism design Network games **Part 2:** Selected papers from leading conferences and journals on the topics that deal with research in mechanismdesign in the paradigm of artificial intelligence and multi-agent systems.
CS718	Sublinear Algorithms For Processing Massive Data Sets		Short Description Algorithmic techniques for processing massive data sets in sublinear space and time, with probabilistic guarantees. Motivation and Goals of the Course There has been a spectacular advance in the capability to acquire data. Applications processing this data have caused a renewed focus on efficiency issues of algorithms. Further, many applications can work with approximate answers and/or with probabilistic guarantees. This opens up the area of design of algorithms that are significantly time and space efficient compared to their exact counterparts. In this course, we will cover modern developments in the area of algorithms for efficient processing of massive data sets. Topics: Compressive sensing. fundamental l1/l2 sparse recovery procedure and universality. Fast sparse recovery: Berinde and Indyk's algorithm. l2/l2 sparse recovery procedures: Algorithms by Gilbert, Li, Porat and Strauss and by Price and Woodruff. Communication Complexity: Introduction and applications: Lower bounds for sparse recovery by Do Ba et.al. and Price and Woodruff. Connection to Kashin's work. l2-Dimensionality Reduction: Johnson-Lindenstrauss's (J-L) Lemma. Fast J-L transforms: Ailon-Chazelle, Dasgupta et.al., Kane and Nelson. Randomized algorithms for matrix problems: Random projections; Matrix multiplication and norm estimation; Random sampling of columns and elements from a matrix; Sampling algorithms for L2 regression and relative error low-rank matrix approximation, Numerical Linear Algebra in streaming model.

			Novel data-motivated matrix factorizations: Sparse PCA; Matrix rank minimization; CUR and related decompositions. (Time permitting) Algorithmic approaches to graph partitioning problems: Flow-based partitioning methods. Spectral-based partitioning methods: Combining spectral and flow-based methods; Local graph partitioning methods. Embeddings and geometric structure related to graphs. Expanders for algorithms and real networks.
CS719	Introduction To Data Streams		Motivating applications: network monitoring, sensor networks, need for highly efficient processing of high speed and high volume data streams, Space and time efficient randomized algorithms as a candidate solution, models of data streams. Basics of randomization: elementary probability theory, expectation, linearity of expectation, variance, Markov and Chebychev's inequality, Chernoff and Hoeffding (CH) tail inequalities, hash functions, limited independence, CH-bounds for limited independence. Finding frequent items in data streams, Estimating distinct item queries, Estimating frequency moments, estimating join sizes, Approximate histograms over data streams, Transforms over data streams, wavelets, fourier and DCT clustering over data streams, Applications to graphs.
CS720	VLSI Testing And Fault-Tolerance		The course is primarily intended to familiarize students with the problem of testing large and complex electronic circuits. Various techniques to solve this problem and concepts of design for easy testability (DFT) will be discussed. Topics related to fault-modeling and fault-simulation to evaluate the fault-coverage of test vectors will be covered in detail. The problem of reduced yield and reliability of circuits in presence of faults will be discussed and techniques to improve the yield and reliability of these circuits by introducing fault-tolerance measures will also be covered. Various redundancy techniques like structural, time, information and software redundancy will also be discussed in detail.
CS724	Sensing, Communications & Networking For Smart Wireless Devices		The course will cover different types of sensing, communication and networking techniques for current/future smart devices. The focus of this course is to build the foundations for building real-world technologies and solutions. The course will start with some mathematical concepts and then how to apply them in solving real-life problems. Key topics of this course include GPS, indoor localization techniques, motion tracking, applications of different sensing modalities, low power wireless protocols etc.

			Tentative Topics **Introduction:** Technology, scope, applications of wireless sensing and sensor networks **Localization:** GPS localization, indoor localization challenges, RSSI based localization, fingerprinting based approaches, Time-of-flight (ToF), Time difference of arrival (TDoA), Clock synchronization **Signal Processing and applications:** Time domain to frequency domain conversion, DFT basics, Beamforming basics and applications, Angle-of-arrival (AoA) based localization **IMU sensor and motion sensing:** Understanding inertial measurement unit (accelerometer, gyroscope, magnetometer), sensor fusion, applications of IMUs for motion tracking, gesture detection, activity tracking etc. **MAC in sensor networks:** Requirements, synchronous vs asynchronous MAC, low-power MAC, specific examples including IEEE 802.15.4 **Routing in sensor networks:** Energy aware routing, geographic routing, attribute based routing etc. **Device-free sensing:** Wireless signals and communication channels for sensing, applications like human presence detection, digital agriculture etc. **Dynamic time warping and applications:** Basics of pattern matching, dynamic time warping, applications like posture detection, hand movement tracking etc.
CS725	Topics In Networking		Recent developments in various fields in networking, including but not limited to, routing, flow control, performance evaluation, transport protocols, application protocols, real-time protocols, and network architectures. Emerging technologies such as, ATM, SMDS, Frame-relay, SONET, ISDN. Issues in Gigabit networking. Network related issues in development of multi-media applications.
CS726	Topics In Multimedia		Multimedia systems - requirements, technology. Coding and compression standards - JPEG, MPEG, etc. Architecture issues in multimedia. Desk area networks. Operating Systems Issues in multimedia - real-time OS issues, synchronization, interrupt handling, etc. Database issuses in multimedia - indexing and storing multimedia data, disk placement, disk scheduling, searching for a multimedia document. Networkng issues in multimedia - Quality-of-service guarantees, resource reservation, traffic

			specification, shaping, and monitoring, admission control, etc. Multicasting issues. Session directories. Protocols for controlling sessions. Security issues in multimedia - digital watermarking, partial encryption schemes for video streams. Multimedia applications - audio and video conferencing, video on demand, voice over IP, etc. Latest developments in the field of multimedia.
CS727	Topics In Internet Technologies		The World Wide Web which started as an application on the Internet has evolved quickly into a platform on which all applications are built. In this course we deal with how to design, build and deploy a contemporary web application. **Perquisites:** Knowledge of: HTTP, HTML, CSS, JavaScript, Server side scripting(Python/Java/PHP..). Familiarity with Cloud Computing will help. **Topics** **Part 1: Fundamentals** Web Applications: Architecture, Application Frameworks Technologies: Cloud computing, Android Apps, Technology Frameworks, MEAN Stack, XML Quality Attributes: Performance, Security **Part 2: Applications** Multilingual Web, Search engines. Semantic Web, Block Chains, Internet of Things
CS728	Topics In Grid Computing		Overview. Focuses on grid computing as emerging new computing paradigm for solving complex collaborative problems that require massive resources and infinite CPU cycle. The topics included: Definition of Grid; Basic Building Blocks; Issues in Management of Grid Models; Evolution of Grid Models. Architecture. Deals with grid architecture providing an anatomical look into fundamental system components and their functionalities as well as interactions. Topics: Requirements concerning abstractions, behaviours, resources, connectivity, and protocols; Open grid service architectures. Environment. Talks about grid computing environments. Topics: Overview of GCE; Programming models; Middleware for building grid computing environments; Language support (MPI-G, MPI-G2, etc) for grid computing; Meta models for grid programming; Security. Applications. Deals with case studies, how the global computing infrastructure has become a reality for collaborative complex data intensive computing aid for federated database services, web services,

			bioinformatics. It will also include among others some selection of topics from Seti project, Sun grid engine, Skyserver and some national grid projects. Monitoring and evaluation. It will include following: Monitoring; Scheduling; Performance tuning; Debugging and performance diagnostic issues;
CS730	Topics In Operating Systems	3-0-0-0 (9)	The course will primarily focus on the following topics. 1. Introduction: Operating System concepts catch-up using Linux kernel as the reference OS. 2. Linux kernel programming: Customizing the Linux kernel, design and implementation of simple kernel modules, and, design of new kernel features with user space interfacing using system call API and character devices. 3. Isolation enforcement in the Linux kernel: Isolation between user and kernel space, process level isolation, process group level isolation techniques (Linux Cgroups), application containers (e.g., docker and LXC), security issues and advanced isolation techniques. 4. OS design for multi-core machines: Synchronization challenges and solutions to address performance and scalability aspects. I/O scalability using hardware enhancements like solid state devices (SSDs), IOMMU, SR-IOV etc. 5. Advanced isolation mechanisms: Design and implementation of virtualized systems, hypervisors (KVM and Xen).
CS731	Blockchain Technology And Applications	**3-0-0-0- [9]**	Blockchain is an emerging technology platform for developing decentralized applications and data storage, over and beyond its role as the technology underlying the cryptocurrencies. The basic tenet of this platform is that it allows one to create a distributed and replicated ledger of events, transactions, and data generated through various IT processes with strong cryptographic guarantees of tamper resistance, immutability, and verifiability. Public blockchain platforms allow us to guarantee these properties with overwhelming probabilities even when untrusted users are participants of distributed applications with ability to transact on the platform. Even though, blockchain technology has become popularly known because of its use in the implementation of Cryptocurrencies such as BitCoin, Ethereum, etc., the technology itself holds much more promise in various areas such as time stamping, logging of critical events in a system,

			recording of transactions, trustworthy e-governance etc. Many researchers are working on many such use cases such as decentralized public key infrastructure, self-sovereign identity management, registry maintenance, health record management, decentralized authentication, decentralized DNS, etc. Also, corporations such as IBM and Microsoft are developing their own applications in diverse fields such as the Internet of Things (IoT), etc., even enabling blockchain platforms on the cloud. Considering the need to disseminate the emerging concepts for students, we decided to prepare the a new course on blockchain technology platforms and applications. The students will be exposed to the following topics: 1. Basic Cryptographic primitives used in Blockchain – Secure, Collison-resistant hash functions, digital signature, public key cryptosystems, zero-knowledge proof systems 2. Basic Distributed System concepts – distributed consensus and atomic broadcast, Byzantine fault-tolerant consensus methods 3. Basic Blockchain (Blockchain 1.0) – concepts germane to Bitcoin and contemporary proof-of-work based consensus mechanisms, operations of Bitcoin blockchain, crypto-currency as application of blockchain technology 4. Blockchain 2.0 – Blockchains with smart contracts and Turing complete blockchain scripting – issues of correctness and verifiability, Ethereum platform and its smart contract mechanism 5. Blockchain 3.0 – Plug-and-play mechanisms for consensus and smart contract evaluation engines, Hyperledger fabric platform 6. Beyond Cryptocurrency – applications of blockchain in cyber security, integrity of information, E-Governance and other contract enforcement mechanisms 7. Limitations of blockchain as a technology, and myths vs. reality of blockchain technology 8. Research directions in Blockchain technology The course will be very heavy on projects and require ability to quickly configure a new

development platform and use it, develop applications, and move to a new one. At least three blockchain platforms will be used in projects in the course. The course will consist of instructor presentations, demonstrations, and hands-on projects.

Module	Topic	No. of 1 hour Lectures
Introduction	Need for Distributed Record Keeping Modeling faults and adversaries Byzantine Generals problem Consensus algorithms and their scalability problems Why Nakamoto Came up with Blockchain based cryptocurrency? Technologies Borrowed in Blockchain – hash pointers, consensus, byzantine fault-tolerant distributed computing, digital cash etc.	3
Basic Distributed Computing	Atomic Broadcast, Consensus, Byzantine Models of fault tolerance	4
Basic Crypto primitives	Hash functions, Puzzle friendly Hash, Collison resistant hash, digital signatures, public key crypto, verifiable random functions, Zero-knowledge systems	4
Blockchain 1.0	Bitcoin blockchain, the challenges, and solutions, proof of work, Proof of stake, alternatives to Bitcoin consensus, Bitcoin scripting language and their use	5
Blockchain 2.0	Ethereum and Smart Contracts, The Turing Completeness of Smart Contract Languages and verification challenges, Using smart contracts to	8

<table>
<tr>
<td></td>
<td></td>
<td></td>
<td>
<table>
<tr><td></td><td>enforce legal contracts, comparing Bitcoin scripting vs. Ethereum Smart Contracts</td><td></td></tr>
<tr><td>Blockchain 3.0</td><td>Hyperledger fabric, the plug and play platform and mechanisms in permissioned blockchain</td><td>8</td></tr>
<tr><td>Privacy, Security issues in Blockchain</td><td>Pseudo-anonymity vs. anonymity, Zcash and Zk-SNARKS for anonymity preservation, attacks on Blockchains – such as Sybil attacks, selfish mining, 51% attacks - -advent of algorand, and Sharding based consensus algorithms to prevent these</td><td>8</td></tr>
<tr><td>Total Lecture hours</td><td></td><td>40 hours</td></tr>
</table>
</td>
</tr>
<tr>
<td>CS733</td>
<td>Topics In Parallel Computing</td>
<td>3-0-0-9</td>
<td>
<table>
<tr><th>S. No</th><th>Broad Title</th><th>Topics</th><th>No.of Lectures</th></tr>
<tr><td>1</td><td>Resource management</td><td>Job scheduling, Slurm, hwloc</td><td>4</td></tr>
<tr><td>2</td><td>Parallel file systems</td><td>Lustre, I/O optimizations, I/O parameter selections</td><td>4</td></tr>
<tr><td>3</td><td>One-sided communications</td><td>Remote memory access, windows</td><td>4</td></tr>
<tr><td>4</td><td>Performance modeling</td><td>computation and communication models, logP, logGP models</td><td>4</td></tr>
<tr><td>5</td><td>Parallel profiling</td><td>Profiling and tracing, understanding popular tools such as TAU, HPCToolkit, I/O profiling using Darshan</td><td>4</td></tr>
<tr><td>6</td><td>Topology-aware mapping</td><td>Mapping heuristics, performance improvement with mapping,</td><td>4</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr><td></td><td></td><td></td><td>
<table>
<tr><td></td><td></td><td>visual representation of topology</td><td></td></tr>
<tr><td>7</td><td>Scientific simulation and visualization</td><td>Simulation, visualization and analysis workflows, large-scale data movement optimizations</td><td>4</td></tr>
<tr><td>8</td><td>Spark and MPI</td><td>Apache Spark, Large-scale data analysis using Spark and MPI</td><td>4</td></tr>
<tr><td>9</td><td>Grid, Cloud, Fog, Edge</td><td>Trends from Grid to Cloud to Fog and Edge</td><td>4</td></tr>
<tr><td>10</td><td>Project review</td><td>Projects based on above topics</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>CS738</td><td>Advanced Compiler Optimizations</td><td></td><td>Introduction to Advanced topics, Compiler Algorithms Notation, Symbol table structure, Intermediate representation, Run time support, Producing code generators automatically, Control flow analysis, Data flow analysis, Dependence analysis and dependence graphs, Alias analysis, Introduction to optimizations, Early optimizations, Redundancy elimination, Loop optimizations, procedure optimizations, Register allocation, Code scheduling, control flow and low level optimizations, Inter procedural Analysis and optimizations, Optimization for memory hierarchy, Case studies.</td></tr>
<tr><td>CS740</td><td>Topics In Logic And Computation</td><td></td><td>Curry-Howard isomorphism between typed terms of formal systems representing computable functions and deductions in certain logics.
Simply typed lambda calculus, Godel's system T and Girard's system F. Strong normalization. Semantics. Expressibility of these -- how higher order functions and polymorphism add to expressibility.
Connection with provably recursive functions in systems of arithmetic. Polynomial time logic.</td></tr>
<tr><td>CS741</td><td>Structural Complexity</td><td></td><td>Survey of basic background: models of computations and resource bounded computations. Central complexity classes, notion of complete problems in a class, polynomial time reducibilities and how these relate to each other.
Structure of NP complete sets, p-isomorphism conjecture. Sparse sets in NP. Self reducibility. Relativised classes. Nonuniform complexity. Uniform diagonalization. Polynomial time hierarchy. Interactive proof systems.</td></tr>
</table>

CS742	Parallel Complexity And Sub-Logarithmic Time Algorithms		VLSI theory of complexity. Theory of log-space completeness. Structure of NC. Unbounded fan-in circuits. CRAM model and allocated PRAM models. Sub-logarithmic time algorithms for Parallel symmetry breaking, parallel prefix computation, ordered chaining, nearest largers, Delaunay triangulation and convex hull. Optimal NC algorithms for deterministic list ranking triconnectivity and task Scheduling.
CS743	Advanced Graph Algorithms		Review of important sequential graph algorithms. Introduction to parallel models for computation. General techniques for fast parallel computations on vectors and lists and their applications to design of efficient parallel graph algorithms. Parallel dynamic programming and its applications to expression graphs. State-of-art algorithms for depth first search of directed and undirected graphs. NC-algorithms for st-numbering and open ear decomposition. Parallel algorithms for graph optimization problems. Algorithms for graph coloring. Decomposition of graph into simpler subgraphs. Equivalence relations and classes in graphs. Parallel planarity testing.
CS744	Pseudo-Random Generators		Pseudo-random generators are efficiently computable functions that stretch an input random string to a much bigger sized string such that the output string appears random to resource-bounded computations. These functions have become one of the fundamental objects to study in complexity theory because of their utility. They are used to derandomize randomized algorithms, formalize notions of cryptographic security, obtain lower bounds on the complexity of problems etc. (unfortunately, as of now very few constructions of pseudo- random generators are provably known although many are conjectured). In this course, we study pseudo-random generators and their connections in depth. The topics covered in the course are as follows: Pseudo-random generators: definitions and Existence. Pseudo-random generators of small stretch: Definitions of cryptographic security, Equivalence of one-way functions and pseudo-random generators, Some functions conjectured to be one-way functions, Pseudo-random function generators.

			Pseudo-random generators of large stretch: Equivalence of lower bounds and pseudo-random generators, Known pseudo-random generators against small depth circuits and small space classes, Extractors and pseudo-random generators. Pseudo-random generators against arithmetic circuits: Equivalence of lower bounds and pseudo-random generators, A function conjectured to be pseudo- random. Other applications.
CS745	Small Space Bounded Computations		In this course we shall study the computational complexity of space bounded computations. In particular, we shall focus on the various models and problems in the logarithmic space domain. We shall see how some of the complexity classes in this setting are characterized by natural computational problems and also study the known relations between these classes. We shall also lookat some of the open questions in this area and discuss potential means to tackle these problems. Topics: 1. Studying the various models of space bounded computations. 2. The complexity classes that arise from these modelsand the known relations between them. 3. Natural computational problems that characterize these classes. 4. Proof techniques in this area such as the double inductive counting technique, pebbling technique, derandomization, etc. 5. Some classical results in this area such as Immerman-Szelepscenyi Theorem, Barrington's Theorem, circuit lower bound for parity, etc. 6. Recent results in this area. 7. Open problems and future research avenues.
CS746	Riemann Hypothesis And Its Applications		Riemann Hypothesis is one of the most important unresolved conjectures in mathematics. It connects the distribution of prime numbers with zeroes of Zeta function, defined on the complex plane. A number of algorithms in algebra and numbertheory rely on the correctness of Riemann Hypothesis or its generalizations. This course willdescribe the connection between prime distributions and Zeta function leading to the Riemann Hypothesis, prove Prime Number Theorem along the way, and then describe the generalizations of Riemann Hypothesis

			and their applications to computer science problems. Topics: 1. Prime counting and other arithmetic functions 2. Brief overview of complex analysis 3. Zeta function definition and basic properties 4. Riemann Hypothesis and its relationship with prime counting 5. Prime Number Theorem 6. Dirichlet L-functions and Generalized Riemann Hypothesis 7. Applications of Riemann and Generalized Riemann Hypothesis 8. Further extensions of Riemann Hypothesis and its proof for function fields, and elliptic curves over finite fields
CS747	Randomized Methods In Computational Complexity		In this course, we will study how randomness helps in designing algorithms and how randomness can be removed from algorithms. We will start by formalizing computation in terms of algorithms and circuits. We will see an example of randomized algorithms, identity testing, and prove that eliminating randomness would require proving hardness results. We then prove a hardness result for the problem of parity using randomized methods. We construct certain graphs called expanders that are useful in reducing randomness in algorithms. These lead to a surprising logarithmic-space algorithm for checking connectivity in graphs. We show that if there is hardness in nature then randomness cannot exist! This we prove by developing pseudo-random generators and error-correcting codes. We show how to extract randomness from a weakly random source by using extractors. Finally, we show how to probabilistically check proofs (PCP) and prove the hardness of approximating some NP-hard problems.
CS748	Arithmetic Circuit Complexity		In this course we will study computation by primarily algebraic models, and use, or in many cases extend, the related tools that mathematics provides. We will start with some positive examples--fast polynomial multiplication, matrix multiplication, determinant, matching, linear/algebraic independence, etc. The related tools are FFT (fast fourier transform), tensor rank, Newton's identity, ABP (algebraic branching program), PIT (polynomial identity testing), Wronskian, Jacobian, etc. One surprising result here is that certain problems for general circuits reduce to depth-3

			circuits. Furthermore, the algorithmic question of PIT is related to proving circuit lower bounds. We then move on to proofs, or attempts to prove, that certain problems are hard and impossible to express as a small circuit (i.e. hard to solve in real life too). One such problem is Permanent. We study the hardness against restricted models-- diagonal circuits, homogeneous depth-3, homogeneous depth-4, noncommutative formulas, multilinear depth-3, multilinear formulas, read-once ABP, etc. The partial derivatives, and the related spaces, of a circuit will be a key tool in these proofs. The holy grail here is the VP/VNP question. Depending on time and interest, other advanced topics could be included. One such growing area is-- GCT (geometric complexity theory) approach to the P/NP question.
CS749			
CS750	Programs, Proof And Types		The course, in its offering next semester, plans to cover programming language foundations in a hands-on way through theorem prover Coq. This simultane- ously covers PL concepts as well theorem proving in Coq. For more details see the link: https://softwarefoundations.cis.upenn.edu/. The course intends to cover most of volume 1 and volume 2 listed on this link. If there is interest and time, we may also cover some other topics. Evaluation Evaluation will be mostly based on theorem proving projects and/or paper presentations.
CS755	Topics In Software Engineering		This is a research-oriented, seminar type course which will focus on the state-of-the-art in various areas of Software Engineering-- Software project management, Metrics and mea surement, Software configuration management, Software risk management, Requirements engineering, Software quality assurance, Software reliability models, Object oriented design, Object oriented programming (with C++), Formal specifications, Formal verification of programs, Jackson method for design, CASE tools and technology, Cleanroom method for software development, Information system design, Real- time software specification and design.
CS771	Introduction To Machine Learning		Machine Learning is the discipline of designing algorithms that allow machines (e.g., a computer) to learn patterns and concepts from data without being explicitly programmed. This course will be an introduction to the design (and some analysis) of machine learning algorithms, with a modern outlook

			focusing on recent advances, and examples of real-world applications of machine learning algorithms. List Of Topics Preliminaries Multivariate calculus: gradient, Hessian, Jacobian, chain rule Linear algebra: determinants, eigenvalues/vectors, SVD Probability theory: conditional probability, marginal probability, Bayes rule Supervised Learning Local/proximity-based methods: nearest-neighbors, decision trees Learning by function approximation Linear models: (multiclass) support vector machines, ridge regression Non-linear models: kernel methods, neural networks (feedforward) Learning by probabilistic modeling Discriminative methods: (multiclass) logistic regression, generalized linear models Generative methods: naive Bayes Unsupervised Learning Discriminative Models:k-means (clustering), PCA (dimensionality reduction) Generative Models Latent variable models: expectation-maximization for learning latent variable models Applications: Gaussian mixture models, probabilistic PCA Practical Aspects Concepts of over-fitting and generalization, bias-variance tradeoffs Model and feature selection using the above concepts Optimization for machine learning: (stochastic/mini-batch) gradient descent Additional Topics (a subset to be covered depending on interest) Deep learning: CNN, RNN, LSTM, autoencoders Structured output prediction: multi-label classification, sequence tagging, ranking Ensemble methods: boosting, bagging, random forests Recommendation systems: ranking methods, collaborative filtering via matrix completion Reinforcement learning and applications Kernel extensions for PCA, clustering, spectral clustering, manifold learning Probability density estimation and anomaly detection Time-series analysis and modeling sequence data Sparse modeling and estimation

			Online learning algorithms: perceptron, Widrow-Hoff, explore-exploit Statistical learning theory: PAC learning, VC dimension, generalization bounds A selection from some other advanced topics such as semi-supervised learning, active learning, inference in graphical models, Bayesian learning and inference
CS772	Probabilistic Machine Learning		Estimating the parameters of the underlying model that is assumed to have generated the data is central to any machine learning problem. Probabilistic modeling offers principled and rigorous ways to model data of diverse types, characteristics, and peculiarities, and offers algorithms to uncover the model parameters and make inferences/predictions about the data. This course will expose the students to the basic concepts/algorithms used in probabilistic modeling of data and we will gradually work our way up to use these as building blocks for solving more complex machine learning problems. We will also, at various points during this course, look at how the probabilistic modeling paradigm naturally connects to other dominant paradigm which is about treating machine learning problems as optimization problems, and understand the strengths/weaknesses of both these paradigms, and how they also complement each other in many ways. A rough outline of the course is given below. 1. Introduction to probabilistic modeling of data 2. Basic methods for parameter estimation in probabilistic models: MLE and MAP estimation 3. Common probability distributions, conjugate priors and exponential family 4. Introduction to Bayesian learning 5. Case studies: Bayesian linear regression and classification, sparse linear models 6. Latent variable models for clustering: mixture models 7. Latent variable models for dimensionality reduction: factor analysis, probabilistic PCA and matrix factorization 8. Latent variable models for modeling sequence and time-series data: hidden Markov models, linear dynamical systems 9. Latent variable models for structured prediction 10. Learning and inference in probabilistic graphical models 11. Approximate Bayesian inference (MCMC, Variational Bayes, Expectation Propagation) 12. Online approximate Bayesian inference

			13. Bayesian learning with kernels: Gaussian Processes (or “Bayesian SVM”) 14. Topic models 15. Deep learning
CS773	Online Learning And Optimization		Instructor’s consent. Familiarity with basics of probability and statistics, and linear algebra would be essential. Prior exposure to machine learning techniques would be desirable. Short Description This is intended to be a course on advanced techniques used in optimization and learning. The techniques introduced in this course can be used to perform tasks such as sequential prediction and optimization on large-scale streaming data. These are the methods of choice for massive learning tasks and form the basis for a large family of optimization and learning routines. The course will be self contained but will nevertheless benefit from familiarity with machine learning as a source of basic learning theoretic concepts, as well as motivation for large-scale learning and optimization. Topics Introduction to preliminaries in convex and real analysis, and probability theory Online Prediction with Full Feedback Online classification – Perceptron, Winnow Online regression – gradient descent, exponentiated gradient Learning with expert advice – weighted majority, hedge Online Convex Optimization Review of batch optimization – batch gradient descent Follow the leader – regularized, perturbed Online gradient descent – polynomial and logarithmic regret bounds Online mirror descent – links to OGD, FTL Stochastic gradient descent – online to batch conversion and application to batch learning Online Prediction with Limited Feedback Stochastic/adversarial multi-armed bandits - EXP3, UCB, Thompson Sampling Linear and contextual bandits - lin-UCB, lin-TS Special Topics (depending on interest and available time) Tracking shifting experts Mini-batch methods Beyond additive regret Online second order methods Derivative-free optimization
CS774	Optimization Techniques		Background: convex analysis, linear and matrix algebra, probability theory

			Preliminaries: applications, optimality and duality conditions First Order Methods Subgradient methods Proximal methods Multiplicative weights update, mirrored descent Second Order Methods Newton method Quasi-Newton methods, L-BFGS Stochastic Optimization Problems Notion of regret, online to batch conversion Methods offering vanishing regret - OGD, EG, OMD Non-convex Optimization Problems Applications - sparse recovery, affine rank minimization, low-rank matrix completion Convex approaches - relaxation-based methods Non-convex approaches - projected gradient descent, alternating minimization Special topics (a subset would be chosen depending on interest and available time) Accelerated first order methods Bayesian methods Coordinate methods Cutting plane methods Interior point methods Optimization methods for deep learning Parallel and distributed methods Robust optimization problems and methods Stochastic mini-batch methods Submodular optimization problems and methods Variance reduced stochastic methods Zeroth order methods
CS775	Topics In Probabilistic Modeling And Inference		A tentative list of topics to be covered in this course includes Fundamentals of probabilistic modeling Basics of probability distributions and their properties Basics of probabilistic inference: MLE/MAP/Bayesian inference Hierarchical modeling, multi-parameter models Bayesian vs frequentist statistics Probabilistic graphical models (directed and undirected models) Probabilistic approaches for linear modeling, Sparse Bayesian Learning Latent variable models Mixture models and latent factor models Latent variable models for dynamic/sequential data Latent variable models for networks and relational data Latent variable models with covariates Approximate Inference Inference in probabilistic graphical models

			MCMC methods Variational methods Scalable inference with stochastic optimization Other methods: Likelihood-free methods, spectral methods, etc. Nonparametric Bayesian methods Gaussian Process for function approximation Dirichlet process and beta processes Other stochastic processes (gamma/point processes, etc., and their applications) Bayesian Optimization Theory of Bayesian statistics Probabilistic programming Other topics based on students' interests Treatment of the above topics will be via several case-studies/running-examples, which include generalized linear models, finite/infinite mixture models, finite/infinite latent factor models, matrix factorization of real/discrete/count data, sparse linear models, linear Gaussian models, linear dynamical systems and time-series models, topic models for text data, etc.
CS776	Deep Learning For Computer Vision		Prerequisites: No formal prerequisite but knowledge of basic probability, calculus and linear algebra is required. The course will make the students familiar with basics of learning-based as well as geometric computer vision. The list of possible topics will be 1. Convolutional Neural Networks 2. AutoEncoders 3. Recurrent Neural Networks 4. Generative Adversarial Networks 5. Camera calibration 6. Epipolar geometry 7. Structure from motion This list will evolve based on the level of students enrolled and their interests. For each of the topics we will start with the basics, touch upon some current applications and then you would be expected to work on an assignment which would have a strong programming component. The course is expected to give you a good foundation if you would like to work on Computer Vision in the future either in academic or industrial research and development.
CS777	Topics In Learning Theory		Pre-Requisites Instructor's consent. Fluency in basic results in probability and statistics would be essential. Prior

			exposure to machine learning or signal processing techniques would be desirable. Short Description This is intended to be a course on advanced techniques used in the design and analysis of machine learning and statistical estimation algorithms. The course is divided into two broad parts, one that primarily looks at the statistical analysis of learning and estimation algorithms, and one that explores a variety of algorithm design techniques currently popular in machine learning and statistics communities. The course will involve an intense application of probabilistic models and techniques and will benefit from prior familiarity with machine learning/signal processing as a source of basic learning theoretic concepts, as well as motivation for large-scale learning and optimization. Topics Statistical Learning Theory Notion of risk, l-risk, Bayes risk, regret, plug-in predictors, regret-transfer bounds Empirical risk minimization, PAC learning, uniform convergence, Glivenko-Cantelli classes Capacity notions - covering numbers, VC/fat-shattering dimension, Rademacher complexity Algorithmic stability, regularized risk minimization Calibrated loss functions and consistency of learning methods Algorithmic Learning Theory Ensemble learning - boosting, random forests Non-convex optimization techniques - gradient descent, alternating minimization, EM Online optimization techniques - learning with experts, online mirrored descent, explore-exploit Fast sampling techniques - hit-and-run, MCMC Additional Topics (depending on interest and available time) Learning with fast rates PAC-Bayes bounds Complex prediction tasks - ranking, structured prediction Negative results - the no-free-lunch and ugly-duckling theorems, hardness of learning Derivative-free optimization
CS779	Statistical Natural Language Processing	3-0-0-0 (09)	Pre-Requisites **Must:** Introduction to Machine Learning (CS771) or equivalent course, Proficiency in Linear Algebra, Probability and Statistics, Proficiency in Python Programming **Desirable:** Probabilistic Machine Learning (CS772), Topics in Probabilistic Modeling and Inference

			(CS775), Deep Learning for Computer Vision (CS776) Level Of The Course Ph.D., M.Tech, and 3rd, 4th year UG Students (7xx level) Course Objectives: Natural language (NL) refers to the language spoken/written by humans. NL is the primary mode of communication for humans. With the growth of the world wide web, data in the form of text has grown exponentially. It calls for the development of algorithms and techniques for processing natural language for the automation and development of intelligent machines. This course will primarily focus on understanding and developing linguistic techniques, statistical learning algorithms and models for processing language. We will have a statistical approach towards natural language processing, wherein we will learn how one could develop natural language understanding models from statistical regularities in large corpora of natural language texts while leveraging linguistics theories. Course Contents Introduction to Natural Language (NL) : why is it hard to process NL, linguistics fundamentals, etc. Language Models: n-grams, smoothing, class-based, brown clustering Sequence Labeling: HMM, MaxEnt, CRFs, related applications of these models e.g. Part of Speech tagging, etc. Parsing: CFG, Lexicalized CFG, PCFGs, Dependency parsing Applications: Named Entity Recognition, Coreference Resolution, text classi cation, toolkits e.g. Spacy, etc. Distributional Semantics: distributional hypothesis, vector space models, etc. Distributed Representations: Neural Networks (NN), Backpropagation, Softmax, Hierarchical Softmax Word Vectors: Feedforward NN, Word2Vec, GloVE, Contextualization (ELMo etc.), Subword information (FastText, etc.) Deep Models : RNNs, LSTMs, Attention, CNNs, applications in language, etc. Sequence to Sequence models : machine translation and other applications Transformers : BERT, transfer learning and applications
CS781	Cognition: Memory		Types of memory, Features (esp. behavioural) of memory, Experimental evidence from: Psychology (human/animal subjects), nerophysiology,

			neurochemistry and imaging, brain lesions and damage in human animals, Models: Schema, Sparse distributed memory, pandemonium, Copy cat, Society of mind, matrix, SAM, TODAM, connectionist.
CS782	Cognitive Semantics		Cognitive Semantics seeks to relate linguistic expressions to conceptual structures in the context of a speech act. The objective of this course is to explore the cognition-language mappings. The course will focus on topics such as: - Conceptual and linguistic structures - Cognition and grammar - Rules and connections - Lexical structure and compositionality - Object, event and relational structures - Spatial and temporal semantics - Speech acts, rhetorical relations, intentionality and implicature - Semantic transference, metonymy and metaphor - Grounding, embodiment, perceptual processes and acquisition - Lexicalization patterns and diachronic processes - Cognitive and linguistic processing in artificial agents and other nonhuman systems - Bilingual and Sign language users
CS783	Visual Recognition	3-0-0-9	In this course we undertake a study of visual recognition from various aspects related to computer vision. Visual recognition encompasses a variety of different tasks, techniques and assumptions. Tasks: The visual tasks could range from instance recognition to human action recognition. In instance recognition, we would be answering specific visual identification questions such as: is this an Airbus A380? Another relevant question is object classification, where we aim to answer questions such as: does this image contain a bike or not? Another relevant task is that of object detection in images and videos: where is the bike in the image? In action recognition we aim at more general tasks such as: what is going on in the video? In the course we will undertake a study of different tasks. Techniques: In terms of techniques, there have been a wide range of machine learning techniques ranging from Adaboost and support vector machines to state of the art deep learning techniques. Many of the machine learning techniques have attained popularity based on their success in visual recognition tasks. Indeed, the success of adaboost for face detection has made boosting popular while deep learning techniques became widely popular once they succeeded in large scale object classification. In this course we aim to understand a

			few of the machine learning techniques involved as applied to visual recognition. Advances: There have been certain assumptions in visual recognition such as the need for large number of manually supervised training samples. While this has been dominant there are a number of techniques that aim to relax this assumption by minimising the need for supervision. These include learning with latent variables, active learning techniques, unsupervised machine learning techniques. In the final part of the course we aim to study these advanced techniques. A brief outline of the topics to be covered in the course are as follows: Introduction to visual recognition Features for visual recognition Object Classification Face Detection/Pedestrian detection/Object Detection Object Segmentation Instance Recognition Deep Learning models for the above tasks Advanced topics like weak supervision, domain adaptation, active learning Unsupervised visual recognition
CS784	Language Acquisition		Part A: Child Language Acquisition: Methodology: Diary studies, Large sample studies, Longitudinal studies. Developmental stages: Prelinguistic development: Onset of perception, Phonological development. Linguistic development: Lexicon, holophrasis, under/overextensions; Morphology and syntax (Grammar), telegraphic speech, compositionality, syntax in lexical development; universals and parameter setting, Discourse organization, deictic reference, discourse dependencies, speech acts and implicature. Bilingual language development: Syntactic and semantic processing; Differentiation, transfer and decay. Non-normal language develop.ment: Phonological, grammatical and pragmatic impairments. Part B: Artificial life, agent based and evolutionary approaches to language acQuisition. Some topics from the following will be discussed. 1. Communicative aspects of language. Origins and use of signs, symbols, words, compositional structures in communication. Origins of a lexicon in multiple agent systems, simulation experiments. Emergence/evolution of syntax in multiple agent systems, simulation experiments. Learning mechanisms and constraints from computational learning theory. Emergence/evolution of language universals. Grounding of linguistic entities.

			Emergence/evolution of intentional and semantic aspects of language. Optimality theory and optimality arguments.
CS785	Multiagent Systems: Games, Algorithms, Evolution		We will explore decision making behaviour in communities of agents and in particular how information is used, produced and transmitted in such situations. We will study game theoretic and evolutionary approaches (with an algorithmic emphasis) to model such behaviour and try to understand what implications the results have for problems in biology and in human and non-human communities. (The course will not be concerned with distributed AI or cooperative problem solving by software agents.) Topics: 1. Biological and human communities - structure and properties. 2. Competition, cooperation, evolution. 3. What is a game? Non evolutionary and evolutionary games. Numerous examples from biology, economics, decision theory, social choice etc. 4. Payoffs, strategies - pure, mixed, behavioural. Stability and equilibrium. 5. Information production, transmission and use in games. Rationality, common knowledge. 6. Competition, non-cooperation and the corresponding games. 7. Cooperation and related games. 8. Dynamic models and evolutionary games. 9. Fairness, trust, reputation in decision making and games. 10. Public goods, the commons and related problems. 11. Brief introduction to mechanism design.
CS786	Computational Cognitive Science		To what degree functions of the mind can be reproduced or simulated by a computer is a question that has become volubly prominent in recent years. It is often posed for public consumption as a matter of *ends* – when will computer performance surpass human performance on interesting challenges. From a scientific standpoint, it is more useful to ask this question from the standpoint of *means* – what resources do organisms have available that let them respond adaptively to situations they encounter in the world? This is the question that computational cognitive scientists seek to answer. Such research frequently starts from parametric characterizations of empirical

			behavior. Theorists then develop computational models that capture the quantitative relationship between these parameters and various experimental conditions. A good empirical fit permits further questions of biological and epistemic plausibility to be asked of the model. Models that pass these quasi-philosophical checks graduate to the status of theories. These accounts are, inevitably, challenged as incomplete or erroneous by further iterations of experiments and models. The cycle of research in cognitive science, therefore, encompasses, in order of the workflow presented above, neuroscience and psychology, statistics, computer science, and philosophy. This course is meant to give both UG and PG students interested in the computational aspects of cognitive science a relatively comprehensive overview of the discipline. Over 4-5 lectures per area, categories of mental phenomena will be introduced via descriptions of empirical studies, followed by the chronology of models seeking to explain them. Instruction in each topic will conclude with an instructor-mediated discussion of the merits of competing models, terminating in an appreciation of promising future directions of research in the area. The instructor's emphasis will slant towards approximate Bayesian approaches to such challenges. We will follow a cycle of continuous evaluation – quizzes will be conducted for each of the 7 topics covered in the course (see below), each counting for 10% of the course grade. The remaining 30% will come from a course project, which will require students to implement, critique and possibly improve upon a state-of-the-art model in one of the 7 areas covered in the course Course Outline Foundations – evidence for invariants in behavior – associativity – Pavlovian conditioning – Minsky, Newell and the strong AI program – the framing problem – production system architectures of the mind – the Bayesian revolution – inference, learning and causation – compositionality and probabilistic programs – approximate computation in the mind – algorithmic accounts of sub-optimal inference Perception – James, Helmholtz, Wundt – classical psychophysics – perceptual modalities – quantification and analysis methods – Gestalt

			principles – assimilation and contrast effects – poverty of stimulus – Gibsonian psychophysics – Anne Treisman's feature integration account – recognition by components – David Knill & Eero Simoncelli's Bayesian visual perception work Memory – early experiments – Miller and the magic number 7 – classical experiment settings and analyses – signal detection theory – Tulving's memory types – Baddeley and the discovery of working memory – Rich Shiffrin's line of models and their problems – Austerweil's random walk model – Standing and the fidelity of visual long-term memory connecting to Tim Brady's recent work – Tom Hills and memory search Decision-making – von-Neumann and *homo economicus* – Rescorla-Wagner, Hall-Pearce and classical conditioning findings – operant conditioning and skill learning – Sutton-Barto-Singh and reinforcement learning building up to skill-learning – cognitive biases in decision-making – Tversky's non-compensatory models – the Gigerenzer fast-and-frugal school of heuristics – fast and slow decisions and their consequences – drift diffusion models and their competition – frugal preference learning Language – semantics and semiotics – neurobiological foundations of language with empirical evidence – language universals and typology – Sapir-Whorf hypothesis, evidence for and against – pragmatics and social signaling – nativist vs emergent models of language learning – Bayesian accounts of structure learning – non-human languages – Wittgenstein and philosophy as language games Motor control and learning – systemic principles, feedback, redundancy, coordination – physiological basis – information processing problems – Peter Dayan's model-free vs model-based motor models – Daniel Wolpert's hierarchical motor control models – Paul Schrater's Bayesian structure learning – hierarchical reinforcement learning – Karl Friston's free energy approach – Nikolai Bernstein's beautiful ideas on the value of noise in the motor system – Jeff Beck's rational sub-optimal inference account Similarity & categorization – Luce, Shepherd and empirical foundations – exemplars vs prototypes debate with empirical data – Nosofsky, Shiffrin and the rise of cluster models – Anderson's rational model – Josh Tenenbaum's Bayesian program –

			hierarchical Dirichlet models of categorization – compositionality and the generation of new categories – Liane Gabora's computational models of creativity
CS789	Special Topics In Languages Acquistion And Origins		Child's acquisition of phonology, lexicon, syntax, semantics, pragmatics, and mappings; Grounding issues in language acquisition; Origin and evolution of speech sounds, lexicon, syntax, semantics, intentional use, Agent and population models, Bootstrapping, Modularity-nonmodularity debate; Critical periods; Rules vs connectionist approaches; Sign language, synchronic and diachronic change, creolization.
CS797	Special Advance Topics In Computer Science		
CS798D	Algorithms For Bayesian Networks And Causality		Pre-Requisites: Basics of Probability and Statistics, Data Structures and Algorithms, or consent of the instructor Course Description: Current machine learning algorithms often deal with datasets having hundred or thousands of features. In the unsupervised learning setting, often such datasets are modelled as random samples from a highdimensional distribution which we are interested to learn. Bayesian Networks are a popular class of probabilistic graphical model that allows us to succinctly represent high-dimensional probability distributions owing to a limited conditional dependence between the component random variables. These networks have been used to model large datasets across several practical topics including Bioinfomatics, Medical Diagnosis, Information Retrieval, Image Processing. In this course, we take up a formal algorithmic study of Bayesian networks. We will look at the central problem of efficiently learning an unknown Bayesian network using an optimal number of samples. Along the way, we will touch upon several information-theoretic tools which are useful in data science. In the second part of the course, we will study Causality using graphical models. Here we consider the probabilistic dependence of each node in the Bayesian network as an independent mechanism subject to manipulation. Specifically, the causal effect of a set of nodes on another is formalized by an intervention that fixes a set of nodes of the Bayes net to particular values ignoring their parents while all other nodes are still governed by the usual probabilistic dependence of the Bayes net. We will discuss several important problems that includes learning causal effects and causal models from observational and interventional data.

			Course Contents: **Title** / **Hours** *Bayesian networks* Revision of Probability Basics — 1 Introduction to probabilistic graphical models — 1 Introduction to Bayesian Networks — 1 Learning a Bayes net using hypothesis selection — 1.5 Information-theoretic Lower bound — 1.5 Fixed-structure discrete Bayes net — 1 Tree-structured discrete Bayes net — 1.5 Fixed structure Gaussian Bayes net — 1.5 NP-Hardness of Learning Bayes net — 1 Learning algorithms: PC and GES — 3 *Causal models* Introduction to Causality — 2 Pearl's formalism and do calculus — 1 Identifiability and complete conditions — 2 Algorithms for Learning Causal Models — 1 Causality and Machine learning — 2 *Additional lectures* Ising models — 3 Gaussian graphical models — 3 Student presentations — 13 **Total** — **39**
CS798F	Introduction To Probability For Computer Science		Course Description: Introduction to Probability for Computer Science. The course intends to cover the basics of probability and then introduce high dimensional probability. In the later part, we consider applications from Computer Science. Course Syllabus: **Basics-0** Random experiments, sample space, sigma-field*, review of sets theory, probability set function*. Mutually exclusive events and probability function, exhaustive events, the inclusion-exclusion formula. Some inequalities: Boole's or union bound and generalization, examples. Probability set function continuity*. Conditional probability, Bayes' theorem, independence of sets of events. **Basics-1** Random Variables/Functions from the sample space to reals, discrete random variables and probability mass function, continuous random variables and cumulative distribution function (cdf), properties of cdf, transformation of variables Y = g(X) and conditions, defining probability density

			function (pdf), examples, transformations of variables. Expectation of random variables, examples from continuous and discrete domains, expectation of function of random variables, linearity of expectation and examples. Basic statistics: expectation, variance, high order expectations, moment generating function, Markov, Chebychev inequalities, exponential moment method, convexity, Jensen's inequality, examples. **Multivariate Distributions** Random vector, joint cumulative distribution function, discrete random vectors and joint probability mass function, continuous cdf and joint pdf, marginal pdfs and pmfs. Expectations, mgf of random vectors, transformations of bivariate random variables, examples. Conditional distributions and expectations, total variance law. Correlation coefficient, independent random variables and connections to product form of joint pdf, joint cdf, product expectation. Extension to multiple random variables, mutual independence, mgfs, simple transformations of multiple random variables, examples. Covariance matrix, matrices of random variables, linear combinations of random variables. **Special Distributions** Bernoulli, Binomial, Rademacher, geometric, multinomial distributions, sampling with and without replacement, Poisson, Gamma, chi-square, beta, normal and multivariate normal distribution and some of its properties. **Tail Probability Concentration bounds** Markov, Chernoff and Hoeffding's bounds, Bernstein bounds, applications, examples, Johnson-Lindenstrauss' Lemma, applications **Possible additional topics Time permitting:** Martingale and techniques based on them, Azuma-Hoeffding bound Lipschitz functions of Gaussian variables Markov chains, stochastic matrices and its properties. Applications/Examples
CS798G	Analysis Of Unconventional Programs	**3-0-0-0 (9)**	Prerequisites The course will expect the students to have a good understanding of the theory of computations (especially computation models, languages, decidability etc.), algorithms (especially complexity analysis, correctness proofs), Discrete Mathematics (especially proof techniques, linear algebra, lattice theory, fixpoint theory), Probability and Statistics. The course will include challenging programming

			assignments and projects, so the students will be expected to be good programmers. Objective Over the last few years, program analysis techniques have been applied to a variety of systems. Many of these cannot be perceived as “conventional” sequential programs written in languages like C, Python or Java. For example, there have been attempts at testing and verifying network routing policies, machine language models, database schemas, concurrent systems, probabilistic/randomized systems and hardware/software co-designs. This course will explore how program analysis techniques have been adapted for such “unconventional” programs. Syllabus Programming Language Fundamentals: syntax and semantics, type systems; Program Analysis of Sequential Programs: control-flow and dataflow analysis, abstract interpretation, deductive verification, fuzzing, symbolic execution; Analysis of concurrent systems: memory consistency models, duductive techniques (like Owicki-Gries), vector-clock based dynamic techniques; Analysis of probabilistic and reandomised systems: syntax and semantics of probabilistic operations, modeling probabilistic distributions, applications (eg. differential privacy, randomized algorithms etc.); Analysis of machine learning systems: relevant properties (robustness, fainess, safety etc.) in the context of ML systems, testing/analysis of ML models; Parallels between program analysis and analysis of hardware designs/co-designs; Analysis of miscellaneous other systems, like quantum programs, network control planes and database schemas.
CS798H	Human-Computer Interaction	3-0-0-0 (9)	What Is The Course About? Daily we come across several computing devices and tools (from smartwatch trackers to search engines). Some of them are a delight to use, and some are crappy and frustrating to learn, understand and use – ever wondered what makes them so? Ever wondered why using a smartphone or learning to code so easy and natural for some people, and so hard for others? Is digital transformation in India going to take off, or no, and why? All these, and more, are the concerns of Human-Computer Interaction (HCI) as a field.

			In this course, we will cover the basic ground for HCI, considering how we can study how people interact with computers (if any) in a given context, how to identify opportunities for improvements to their work, and how to systematically design systems that are a delight. The field itself is interdisciplinary and borrows theories and methods from various fields including psychology, computer science, design, sociology and anthropology. This course will give a flavor of this interdisciplinary nature, as well as introduce you to these methods and theories that will come in handy in a lot of contexts. We will also consider some common contexts in which Human-Computer interaction is tricky, what we can do about it, and where the open problems lie! Who Should Take It? If you are looking to get a job as an interaction designer, user designer, or any form of design job – this course would be required. If you are a researcher in CS and are looking to understand how people will use the stuff you build, then you should take this. There is a human aspect to pretty much every subfield of computer science—and a lot of work is publishable in all major CS conferences (Check yours!). If you are a student in CGS or Design, you will find the material very relevant, and offering a complementary set of perspectives than you normally have. If you intend to start up, this course is a must-have, because design is a key differentiator between competing products. Anyone else who is curious, wants to better understand the world we cohabit with computers, and exercise their creative brain muscles – this will be a fun course! What Is The Course Going To Be Like? HCI is an “applied” discipline, much like software engineering or design. So expect to do a bunch of hands-on work—in class, and as part of projects and assignments. There will be 3 hours of lectures (2 lectures x 1.5 hours long). Of course there will breaks, videos, etc! There will be weekly assignments and readings. This will include paper and design critiques, reading textbook chapters, designing studies, observing people, etc. There will be a group project, running across most of the term. You will find user needs / problems with an existing piece of software artifact, design/redesign it, evaluate the design. At the end of the course, you will present all of this + write a

			paper. There will be periodic milestones building up to the final version! There will be bi-weekly quizzes. What Will You Cover? **Chapter-1:** Introduction to HCI. What is HCI – Interdisciplinary nature of HCI – History of HCI – Importance of design and HCI – The design process -- The double diamond of design. **Chapter-2:** Need finding. Data collection (interviews, surveys, observational studies, contextual inquiry). Sampling. Qualitative data analysis (coding, thematic analysis/card sorting, focus group, inter-rater reliability, threads to validity, triangulation). Quantitative data analysis in need finding. Ethical considerations in human studies. **Chapter-3:** Models, theories and frameworks in HCI. What are they and why are they useful? Perception and memory. Model human processor. Information foraging. Activity theory. Distributed cognition. Mental models. **Chapter-4:** Prototyping (low-fi and hi-fi prototypes, use of metaphors). Evaluation (quantitative – controlled experiments, measures, statistical comparisons; qualitative – heuristics, cognitive walkthroughs, desirability and reaction toolkits, hallway usability study). **Chapter-5:** Recurring problems in HCI and ideas on how to approach them: Human-information interaction (search and browsing, information seeking and sensemaking, minimal learning); Human-AI interaction (role of the AI agent, challenges of non-determinism, explanation, trust, principles, wizard of oz methods); Human-robot interaction (Verbal and non-verbal interaction, embodiment, emotions, social robots and robots for autism); HCI for creativity support (nature of creative tasks and supporting them, focus and interruptions, cognitive load, case of programming); HCI for development (the minimum common denominator, constraints, the next billion users, cultural aspects); HCI for collaboration (social mechanisms in collab, facilitating communication and coordination, awareness and catching up, personal vs. professional) HCI for learning and education (if time permits). Inclusive design (if time permits)
CS799	PhD Thesis		
CS899	MS Thesis		
CS888	Introduction to Profession and Communication	2-0-0-0 (6)	**a. Objectives:** The course will skill students on effective research communication as well as

			introduce them to some commonly used research methodologies and paper-writing techniques used in various sub-areas of computer science including theory, systems and data science. **b. Logistics:** The course will be offered once every academic year by a team of 4 faculty members, one of whom will assume the position of instructor-in-charge and the other three members representing the areas of theory, systems and data science. All PhD students will be required to register for this course in their second year. **c. Content:** The course will have 5 parts ***Part 1 (Communication):*** students will be introduced to effective research communication. Specifically, the students will be given general guidelines on preparing posters and oral presentations and writing a research paper. The students may be asked to prepare a short poster, presentation and write-up followed by a discussion on these topics that highlight general communication guidelines such as grammatical correctness, precision, effective use of images and the importance of legible figures, organization, flow of argument, pace of presentation, and others. ***Part 2 (Research Methodology and Paper writing in Theory):*** Introduction to proof techniques (deduction, induction, contrapositive, contradiction, diagonalization) with some examples, power of randomization in algorithm design, notions of intractability, and how hardness can be a boon. The difference between Conjecture, Axiom, Hypothesis and Theorem; The difference between an efficient algorithm and an efficient heuristic. How to survey the literature, do research, and prove something new and publicationworthy. How to structure a theory paper and make it readable for non-experts. ***Part 3 (Research Methodology and Paper writing in Systems):*** Overview of different dimensions of system research: A. Problem formulation — Building/evolving hypotheses and problem statements, motivating problems and proposed solution ideas. B. Empirical analysis — experiment design, analysis, reporting, tools and techniques C. Design and implementation — Root cause analysis and development involving existing systems, tools and techniques for navigating large

			code bases, integration of proposed ideas, checking correctness, bug fixing approaches and tools. ***Part 4 (Research Methodology and Paper writing in Data Science):*** Much of the theoretical and empirical aspects of data science are already covered in the earlier two parts. This part will additionally introduce students to more specialized aspects relevant for data science research pipelines such as: A. Typical research design: problem definition, mathematical solution, algorithmic implementation, evaluation. B. Considerations for evaluation: cross-validation, leaderboards, data leakage, hypothesis testing, dataset decay. ***Part 5 (Hands-on Application):*** This part will require students to prepare an extended abstract (around 4 pages) for a research paper. **d. Evaluation:** An S/X grade will be awarded based on satisfactory completion of assignments and preparation of an extended abstract for a research paper. **Tentative breakup of Lectures:** Topics in the course may he shifted across parts as deemed fit by the instructors. For each of the first four parts, assignments would be provided to help students appreciate research methodologies and communication skills. For instance, the 1st lecture of the week may conclude with an assignment, and the 2nd lecture of the week may focus on discussing that assignment. A suggestive breakup of lectures for each part is given below with parts 1 and 5 being overseen by the instructor in-charge and the rest being overseen by instructors representing the areas of theory, systems, and data science. Part 1: Week 1-3 (6 Lectures) Part 2: Week 4-6 (6 Lectures) Part 3: Week 7-9 (6 Lectures) Part 4. Week 10-11 (4 Lectures) Part 5: Week 12-13 (4 Lectures) **Short summary for inclusion in the Courses of Study booklet:** the course is intended to infroduce CSE PhD students to effective research communication and common research methodologies and paper-writing technigues used in theory, systems, and data science.
CS714	Secure Computation	3-0-0-0- [9]	**Prerequisite:** CS201, CS202, and CS203 or equivalent. CS 641 or equivalent would be useful,

			but not necessary. C8670 would be useful, but not necessary. The instructor will try making the course as self-contained as possible. Main prerequisite is mathematical maturity. **Who can take the course:** Ph.D., Masters, 3rd and 4th year UG Students **Departments that may be interested:** CSE, Mathematics and Statistics Course Rationale: The early pioneers of the Internet envisioned it would be the great equalizer and lead to the complete democratization of the online world—nearly four decades since the reality has diverged considerably from their grand vision. Rather than the bastion of freedom and free speech, which the early visionaries envisioned, the Internet has become a dangerous place. Data theft leading to identity theft has become commonplace. Despite these challenges, it is undeniable that the Internet has revolutionized our lives by making day-to-day tasks easier and connecting people worldwide — however, the risks associated with the Internet loom large. Secure Computation is a promising technique that allows users to keep their data private without sacrificing the Internet's benefits. Interestingly, the origins of Secure Computation were during the exact times when the Internet was in its nascent stages. The early researchers of Secure Computation studied it mainly as a theoretical endeavor. However, nearly four decades after its inception, Secure Computation is more than just a problem of theoretical importance; it can solve practical privacy problems many of which arise due to the expansion of the Internet. **Course Objectives:** On completion of this course, a student should be able to: (i) articulate the definition of Secure Multiparty Computation (i} articulate different MPC constructions, prove their security and correctness; (iii) articulate the definitions of Oblivious Random Access; (iv) articulate the construction of different types of Oblivious RAM protocols; (v) build cryptographically secure systems using Secure Computation.
CS801	Innovations in Computer Science and Engineering	(0-0-0-0) (S/X mode)	**Objectives:** The PG seminar course will expose students to research directions being pursued by various groups within the department and help them make an informed choice of thesis topic. The course will also give them an opportunity to develop technical presentation skills by presenting their work.

			Logistics: The PG seminar course will be offered once every academic year with the DPGC convener and one other faculty member as co-instructors. Al PG students (MTech, MS, PhD) will be required to register for this course in their first year. Additionally, PhD students will be required to register for this course a second time in their 3rd year. **Content:** The course will have 2 parts i. Weekly seminars: PG students along with their supervisor(s) will make presentations on research directions being pursued by their group. ii. Research colloquium: PG students will present their past and ongoing work. PhD students registering for the course a second time will be required to present their work at the colloquium although others will be welcome too. **Evaluation:** PG students registering for the first time will be evaluated on the basis of attendance at the seminars and colloquium. PhD students registering for a second time will be evaluated on the basis of presentation at the research colloguium. Short summary for inclusion in the Courses of Study booklet: the course is intended to introduce CSE PG students to cutting edge research directions as well as give them a chance to develop presentation skills by presenting work done by them/their research group to their peers and others.