

## COMPUTER SCIENCE & ENGINEERING (CSE)

### *Undergraduate Courses*

**099. Undergraduate Research/Independent Study. (C)** A maximum of 2 c.u. of CSE 099 may be applied toward the B.A.S. or B.S.E. degree requirements.

An opportunity for the student to become closely associated with a professor (1) in a research effort to develop research skills and techniques and/or (2) to develop a program of independent in-depth study in a subject area in which the professor and student have a common interest. The challenge of the task undertaken must be consistent with the student's academic level. To register for this course, the student must submit a detailed proposal, signed by the independent study supervisor, to the SEAS Office of Academic Programs (111 Towne) no later than the end of the "add" period.

### **L/R 110. Introduction to Computer Programming (with Java, for beginners). (C)**

How do you program computers to accomplish tasks? How do you break down a complex task into simpler ones? CSE 110 is a "Java lite" course that covers the fundamentals of object-oriented programming such as objects, classes, state, methods, loops, arrays, inheritance, and recursion using the Java programming language.

### **112. (PPE 112) Networked Life. (C)**

How does Google find what you're looking for... and exactly how do they make money doing so? What properties might we expect any social network (such as the Penn Facebook) to reliably have, and are there "simple" explanations for them? How does your position in a social or economic network (dis)advantage you, and why? What might we mean by the economics of spam? What do game theory and the Paris subway have to do with Internet routing? Networked Life looks at how our world is connected -- socially, economically, strategically and technologically -- and why it matters.

### **L/R 120. Programming Languages and Techniques I. (C)**

This will be a fast-paced introduction to the fundamental concepts of programming, with Java as the main experimental vehicle. We assume some previous programming experience at the level of a high school computer science class. If you got at least 4 in the AP Computer Science A or AB exam, you will do great. However, we do not assume you know Java. Basic experience with any programming language (for instance C, C++, VB, PHP, Perl, or Scheme) will be sufficient. A quiz will be given in the second week of class to test your programming knowledge so that you can decide whether the class is for you. If you have never programmed before, you should take CIS 110 first. We will mainly use Java and the DrJava programming environment, but we will also experiment with Python, a higher-level language.

**L/R 121. Programming Languages and Techniques II. (B)** Prerequisite(s): CIS 120, CIS 260 is a pre or co-requisite for but will be strictly a pre-requisite effective Fall 2009.

This is a course about Algorithms and Data Structures using the JAVA programming language. We introduce the basic concepts about complexity of an algorithm and methods on how to compute the running time of algorithms. Then, we describe data structures like stacks, queues, maps, trees, and graphs, and we construct efficient algorithms based on these representations. The course builds upon existing implementations of basic data structures in JAVA and extends them for the structures like trees, studying the performance of operations on such structures, and their efficiency when used in real-world applications. A large project introducing students to the challenges of software engineering concludes the course.

### **125. (EAS 125) Technology and Policy.**

Have you ever wondered why sharing music and video generates such political and legal controversies? Is information on your PC safe and should law enforcement be able to access information you enter on the Web? Will new devices allow tracking of your every move and every purchase? CIS 125 is focused on developing an understanding of existing and emerging technologies, along with the political, societal and economic impacts of those technologies. The technologies are spread across a number of engineering areas and each of them raise issues that are of current concern or are likely to be a future issue.

### **140. (COGS001, LING105, PHIL044, PPE 140, PSYC107) Introduction to Cognitive Science. (A)**

Prerequisite(s): An introductory course in Computer Science, Linguistics, Neuroscience, Philosophy or Psychology.

How do minds work? This course surveys a wide range of answers to this question from the disciplines ranging from philosophy to neuroscience. The course devotes special attention to the use of simple computational and mathematical models. Topics include perception, action, thought, learning, memory and social interaction.

**240. Introduction to Computer Architecture. (A)** Prerequisite(s): CIS 110 or equivalent experience.

You know how to program, but do you know how computers really work? How do millions of transistors come together to form a complete computing system? This bottom-up course begins with transistors and simple computer hardware structures, continues with low-level programming using primitive machine instructions, and finishes with an introduction to all aspects of computer systems architecture and serves as the foundation for subsequent computer systems courses, such as Digital Systems Organization and Design (CIS 371), Computer Operating Systems (CIS 380), and Compilers and Interpreters (CIS 341).

The course will consider the SPARC architecture, boolean logic, number systems, and computer arithmetic; macro assembly language programming and subroutine linkages; the operating system interface and input/output; understanding the output of the C compiler; the use of the C programming language to generate specific assembly language instructions.

**L/R 260. Mathematical Foundations of Computer Science. (B)**

What are the basic mathematical concepts and techniques needed in computer science? This course provides an introduction to proof principles and logics, functions and relations, induction principles, combinatorics and graph theory, as well as a rigorous grounding in writing and reading mathematical proofs.

**L/R 261. Discrete Probability, Stochastic Processes, and Statistical Inference. (B)** Prerequisite(s): CSE 260 or equivalent.

This course tightly integrates the theory and applications of discrete probability, discrete stochastic processes, and discrete statistical inference in the study of computer science. The course will introduce the Minimum Description Length Paradigm to unite basic ideas about randomness, inference and computation. Students will be expected to use the Maple programming environment in homework exercises which will include numerical and symbolic computations, simulations, and graphical displays.

**L/R 262. Automata, Computability, and Complexity. (A)** Prerequisite(s): CSE 260.

The course provides an introduction to the theory of computation. The treatment is mathematical, but the point of view is that of Computer Science. Broadly speaking, the theory of computation consists of three overlapping subareas: (1) formal languages and automata; (2) computability and recursive function theory; (3) complexity theory. The course will focus mostly on (1) and (2). The topics covered include finite automata and regular languages, context-free languages, Turing machines, Church's Thesis, undecidability, reducibility and completeness, time complexity and NP-completeness.

**277. Introduction to Computer Graphics Techniques. (C)** Prerequisite(s): CIS 120.

This course is focused on programming the essential geometric and mathematical concepts underlying modern computer graphics. Using 2D and 3D implementations, it covers fundamental topics on scene graphs, computational geometry, graphics algorithms, and user interface design. Programming languages introduced include C++, OpenGL, FLTK and Python.

**320. Introduction to Algorithms. (B)** Prerequisite(s): CSE 120, 121, 260, 262.

How do you optimally encode a text file? How do you find shortest paths in a map? How do you design a communication network? How do you route data in a network? What are the limits of efficient computation? This course gives a comprehensive introduction to design and analysis of algorithms, and answers along the way to these and many other interesting computational questions. You will learn about problem-solving; advanced data structures such as universal hashing and red-black trees; advanced design and analysis techniques such as dynamic programming and amortized analysis; graph algorithms such as minimum spanning trees and network flows; NP-completeness theory; and approximation algorithms.

**330. Design Principles of Information Systems. (A)** Prerequisite(s): CSE 121 and 260.

Introduction to database management systems and principles of design. The Entity-Relationship model as a modeling tool. The relational model: formal languages, the industry standard SQL, relational design theory, query optimization. Storing and querying XML data. Datalog and recursive queries. Views and data integration. Overview of system level issues: physical data organization, indexing techniques, and transactions. Connecting databases to the Web. Course work requires programming in several different query languages, several written homeworks and a team project.

**334. Advanced Topics in Algorithms. (M)** Prerequisite(s): CSE 320.

Can you check if two large documents are identical by examining a small number of bits? Can you verify that a program has correctly computed a function without ever computing the function? Can students compute the average score on an exam without ever revealing their scores to each other? Can you be convinced of the correctness of an assertion without ever seeing the proof? The answer to all these questions is in the affirmative provided we allow the use of randomization. Over the past few decades, randomization has emerged as a powerful resource in algorithm

design. This course would focus on powerful general techniques for designing randomized algorithms as well as specific representative applications in various domains, including approximation algorithms, cryptography and number theory, data structure design, online algorithms, and parallel and distributed computation.

**340. Problem Solving and Programming. (M)** Prerequisite(s): CSE 120, 121.

This course is about the principles of programming languages. It studies programming language concepts by implementing a sequence of interpreters, compilers, and type checkers, each one introducing a new language concept. The goal of this course is threefold: By studying the concepts and abstractions of high-level programming languages, students should be able to use them more effectively. Second, by learning how the features of high-level programming languages are implemented, students should be able to program more expressively in low-level languages. Finally, by understanding the principles behind programming language design, students should be able to create, evaluate and compare programming languages.

**341. Compilers and Interpreters. (M)** Prerequisite(s): Two semesters of programming courses, e.g., CSE 120-121, and CSE 240.

You know how to program, but do you know how to write programs that understand and generate other programs? This is the focus of CSE 341. In addition to traditional programming language implementation topics (such as lexing, parsing, grammars, symbol tables, code generation, optimization, garbage collection, and object-oriented implementation), this course also explores the more general problem of reasoning about computation (e.g., for detecting bugs or security constraint violations). CSE 341 includes a substantial and rewarding Java programming project to develop a fully operational compiler for a Java-like object oriented programming language.

**350. Software Design/Engineering. (M)** Prerequisite(s): CSE 240.

Large systems versus small programs. Problems of scale. Software life-cycle: design phase, implementation phase, testing, maintenance. Software re-use. Tools/Toolkits/Libraries. Programming as a group activity. Support tools, e.g., SCCS and RCS. Standards. Software readability and structure. Reading code. Style sheets. Software Testing: role in process, test cases, testers. Documentation. Embedded documentation and external documentation.

**371. Computer Organization and Design Lab. (B)** Prerequisite(s): CIS 240.

This is the second computer organization course and focuses on computer hardware design. Topics covered are: (1) basic digital system design including finite state machines, (2) instruction set design and simple RISC assembly programming, (3) quantitative evaluation of computer performance, (4) circuits for integer and floating-point arithmetic, (5) datapath and control, (6) micro-programming, (7) pipelining, (8) storage hierarchy and virtual memory, (9) input/output, (10) different forms of parallelism including instruction level parallelism, data-level parallelism using both vectors and message-passing multi-processors, and thread-level parallelism using shared memory multiprocessors. Basic cache coherence and synchronization.

**372. Computer Organization and Design Lab. (B)** Corequisite(s): CSE 371.

Laboratory for CSE 371. In this laboratory section, students gain experience with digital design techniques by designing and implementing actual circuits using Verilog HDL and FPGAs. Five assignments culminate in the design and simulation of a complete 16-bit integer pipelined CPU.

**380. Computer Operating Systems. (A)** Prerequisite(s): CSE 240 or EE 300.

This course surveys methods and algorithms used in modern operating systems. Concurrent distributed operation is emphasized. The main topics covered are as follows: process synchronization; interprocess communication; concurrent/distributed programming languages; resource allocation and deadlock; virtual memory; protection and security; distributed operation; distributed data; performance evaluation.

**381. Computer Operating System Lab. (A)** Corequisite(s): CSE 380.

This course is a semester long project to design and implement your own operating system. Typical components include a process management system, a command interpreter, and a file management system.

**390. (MEAM420, MEAM520) Machine Perception. (M)** Prerequisite(s): MATH 240, PHYS 150 or MEAM 110/147.

The rapidly evolving field of robotics includes systems designed to replace, assist, or even entertain humans in a wide variety of tasks. Recent examples include planetary rovers, robotic pets, medical surgical-assistive devices, and semi-autonomous search-and-rescue vehicles. This introductory-level course presents the fundamental kinematic, dynamic, and computational principles underlying most modern robotic systems. The main topics of the course include: coordinate transformations, manipulator kinematics, mobile-robot kinematics, actuation and sensing, feedback control, vision, motion planning, and learning. The material is reinforced with hands-on lab exercises including basic robot-arm control and the programming of vision-guided mobile robots.

**391. Introduction to Artificial Intelligence. (M)** Prerequisite(s): CSE 121 and CSE 262; CSE 341 strongly recommended.

Artificial Intelligence is considered from the point of view of a resource--limited knowledge-based agent who must reason and act in the world. Topics include logic, automatic theorem proving, search, knowledge representation and reasoning, natural language processing, probabilistic reasoning, and machine learning. Programming assignments in Prolog and C++ or Java.

**398. Quantum Computer and Information Science. (C)** Prerequisite(s): CSE 260, 262 and Math 240.

The purpose of this course is to introduce undergraduate students in computer science and engineering to quantum computers (QC) and quantum information science (QIS). This course is meant primarily for juniors and seniors in CSE. No prior knowledge of quantum mechanics (QM) is assumed. Enrollment is by permission of the instructor.

**400. Senior Project. (A)** Prerequisite(s): Senior standing or permission of instructor.

The goal of the senior design course is to provide students with an opportunity to define, design and execute a significant project. Project subjects may revolve around software, hardware or computational theory. Students must have an abstract of their Senior Project, which is approved and signed by a Project Advisor early in the Fall semester. The project is expected to span two semesters; students must enroll in CSE 401 during the second semester. At the end of the first semester, students are required to submit an intermediate report and give a presentation describing their project and progress. Grades are based on technical writing skills (as per submitted report) presentation skills and progress on the project. These are evaluated by the Project Adviser and the Course Instructor.

**401. Senior Project. (B)** Prerequisite(s): CSE 400, senior standing or permission of instructor.

Continuation of CSE 400. Design and implementation of a significant piece of work: software, hardware or theory. Students are required to submit a final written report and give a final presentation and demonstration of their project. Grades are based on the report, the presentation and the satisfactory completion of the project. These are evaluated by the Project Adviser and the Course Instructor.

**410. (CIS 510) Curves and Surfaces: Theory and Applications. (M)**

The course introduces mathematical and algorithmic techniques for geometric modeling with applications to computer graphics and computer animation. The course covers implicit and parametric curves; implicit and parametric surfaces; polygonal surfaces; polygonal surface simplification, decomposition, and parametrization; and surface reconstruction from point sets.

**430. Introduction to Human Language Technology. (A)** Prerequisite(s): CIS 121.

Automatic summarization can help alleviate the information overload problem caused by the unprecedented amount of online textual information. The building of a summarization system requires good understanding of the properties of human language and the use of various natural language tools. In this course we will build several summarization systems of increasing complexity and sophistication. In the process we will learn about various natural language processing tools and resources such as part of speech tagging, chunking, parsing, Wordnet, and machine learning toolkits. We will also cover probability and statistics concepts used in summarization, but also applicable to a wide range of other language-related tasks.

**434. (CIS 534) Introduction to Parallel Processing. (C)**

This course is a pragmatic introduction to parallel and distributed programming. It targets science and engineering students with basic programming skills, and prepares them for parallelizing existing sequential programs or optimizing the performance of existing parallel codes. The course teaches how to program with widely used parallel programming interfaces such as Pthread, MPI, OpenMP, HPF and RMI. In addition, the course covers enough information on common parallel architectures, so that the students can optimize the programs for different platforms.

**455. (CIS 555) Internet and Web Systems. (C)** Prerequisite(s): CIS 330, CIS 380 recommended.

This course focuses on Internet and Web technologies and the underlying principles of distributed systems, information retrieval, and data management. The material covered will include web and application server architectures, SML and semistructured data, schema mediation, document indexing and retrieval, peer-to-peer systems, distributed transactions and remote procedure calls. The course has a substantial group implementation project.

**460. (CIS 560) Computer Graphics. (A)** Prerequisite(s): One year programming experience (C, JAVA, C++).

A thorough introduction to computer graphics techniques, covering primarily 3D modeling and image synthesis. Topics cover: geometric transformations, geometric algorithms, software systems (OpenGL), 3D object models (surface and volume), visible surface algorithms, image synthesis, shading and mapping, ray tracing, radiosity, global illumination, photon mapping, anti-aliasing and compositing.

**461. (CIS 561) Computer Modeling and Animation Applications. (C)** Prerequisite(s): CSE 120, 121 or equivalent experience and concurrent or past enrollment in CSE 460/560.

This project-based course is designed to provide a comprehensive introduction to the application of computer graphics in a laboratory setting. Course materials and labs will facilitate understanding issues and trends in 3D computer graphics. Students will develop a facility with fundamental 3-D models and modeling software through a series of projects. The course will offer students a technical understanding of Polygonal and Spline based modeling, alternative and standard methods of 3-D model import and export, and model conversion. It will also cover procedural and scripting methods, techniques, and conventions for creating models and shaders that will function properly for rendering and animation. Practical application of topics covered in CSE 460/560 include; geometric transformations, hierarchies, articulation, modeling, blend shapes, vertex weighting, and animation. Experiments with various animation methods include: dynamics, forward and inverse kinematics, surface deformations, keyframe interpolation, motion capture, procedural animation, and facial animation. The course will be laboratory based and will use industry standard software.

**462. (CIS 562) Computer Animation. (C)** Prerequisite(s): Previous exposure to major concepts in linear algebra (i.e. vector matrix math), curves and surfaces, dynamical systems (e.g. 2nd order mass-spring-damper systems) and 3D computer graphics has also been assumed in the preparation of the course materials.

This course covers core subject matter common to the fields of robotics, character animation and embodied intelligent agents. The intent of the course is to provide the student with a solid technical foundation for developing, animating and controlling articulated systems used in interactive computer games, virtual reality simulations and high-end animation applications. The course balances theory with practice by "looking under the hood" of current animation systems and authoring tools and examines the technologies and techniques used from both a computer science and engineering perspective. Topics covered include: geometric coordinate systems and transformations; quaternions; parametric curves and surfaces; forward and inverse kinematics; dynamic systems and control; computer simulation; keyframe, motion capture and procedural animation; behavior-based animation and control; facial animation; smart characters and intelligent agents.

**477. (LING549) Mathematical Methods/Techniques for Linguistics and Natural Language Processing. (M)** Prerequisite(s): PHIL 006 or instructor's permission.

Basic concepts of set theory, relations and functions, properties of relations. Basic concepts of algebra. Grammars, languages, and automata- finite state grammars, regular expressions, context-free and context-sensitive grammars, unrestricted grammars, finite automata, pushdown automata and other related automata, Turing machines, Syntax and semantics of grammar formalisms. Strong generative capacity of grammars, Grammars as deductive systems, parsing as deduction. Relevance of formal grammars to modeling biological sequences. The course will deal with these topics in a very basic and introductory manner--ideas of proofs and not detailed proofs, and more importantly with plenty of linguistic examples to bring out the linguistic relevance of these topics.

The course will deal with these topics in a very basic and introductory manner--ideas of proofs and not detailed proofs, and more importantly with plenty of linguistic examples to bring out the linguistic relevance of these topics.

**480. Real-Time and Embedded Systems. (M)** Prerequisite(s): CIS 380, some network programming experience is desirable.

Ever increasing availability of inexpensive processors connected by a communication network has motivated the development of numerous concepts and paradigms for distributed real-time embedded systems. The primary objectives of this course are to study the principles and concepts of real-time embedded computing and to provide students hands-on experience in developing embedded applications. This course covers the concepts and theory necessary to understand and program embedded real-time systems. This includes concepts and theory for real-time system design, analysis, and certification; programming and operating systems for embedded systems; and concepts, technologies, and protocols for distributed embedded real-time systems.

The course will cover a variety of existing systems and technologies, e.g., real-time machines, architectural description language, formal method and logical-time programming paradigms, and certification. The course requires active student participation in-group projects. Each group will be responsible for the design and implementation of a life-critical embedded system such as a pacemaker. The group projects are intended to complement the learning of principles and concepts through the application of theory in practice and the development of experimental skills in building embedded applications.

**482. (CIS 582) Logic In Computer Science. (C)** Prerequisite(s): CSE 260.

Logic has been called the calculus of computer science as it plays a fundamental role in computer science, similar to that played by calculus in the physical sciences and traditional engineering disciplines. Indeed, logic is useful in areas of computer science as disparate as architecture (logic gates), software engineering (specification and verification), programming languages (semantics, logic programming), databases (relational algebra and SQL), artificial intelligence (automatic theorem proving), algorithms (complexity and expressiveness), and theory of computation (general notions

of computability). CSE 482 provides the students with a thorough introduction to mathematical logic, covering in depth the topics of syntax, semantics, decision procedures, formal proof systems, and soundness and completeness for both propositional and first-order logic. The material is taught from a computer science perspective, with an emphasis on algorithms, computational complexity, and tools. Projects will focus on problems in circuit design, specification and analysis and protocols, and query evaluation in databases.