CS491
Great Principles of Information Technology

Prepared by P. J. Denning

1/22/01
Approved by CS Faculty as new course
and as senior option course

I. CATALOG INFORMATION

A. CS 491: Great Principles of Information Technology (3:3:0)

B. Prerequisite: senior standing (at least 90 credit hours) including two 400-level CS courses.

C. Catalog Description: A synthesis course for CS majors. Offers a holistic view of the field and its connections with other fields. Covers great principles of information technology from algorithms and programming, distributed systems, and cooperative systems. Emphasizes the historical development of these principles, why they have stood the tests of time, how they relate to one another, and how they relate to issues in other fields. Also covers major contemporary open questions in information technology. Includes a project with an oral presentation.

D. Course Objectives: This course aims to develop students' sense of the whole of information technology by synthesizing and integrating their existing knowledge from separate CS specialties. It discusses the great principles on which CS is founded, their relationships with one another, and their relationships with other fields. It examines what makes a principle great -- durable, wide, and lasting impact -- and traces the history of each principle to demonstrate why it is great. It also examines several of the major system problems facing information technologies today, speculating about how the great principles will influence the development of solutions and what new principles might emerge from the development.

E. Logistics of Student Presentations: The course includes student teams (nominal size 3) working on projects with a final oral presentation to the instructor (or faculty team). It is easy to do this without increasing the workload of the course. In a class of size 39, there would be 13 team presentations rather then 39 individual presentations. An appropriate amount of time would be build into the schedule to allow for this. Denning does this routinely in CS471 without difficulty.

II. SYLLABUS

NOTE: the selection of topics within each category listed below will vary according to the expertise of the participating faculty or guest lecturers. This framework will encourage and accommodate differing perspectives of the participating faculty.

Principles, Ideas, and Greatness (1 week)

A principle is a law or a statement of a condition from which many other conditions can be deduced. A idea is a concept, abstraction, or representation of an entity. The course topics are grouped by classes of related ideas (areas of coherent abstractions) and are explained in terms of a few principles in each category. The criterion of greatness means: (1) The principle consistently solves important problems that vexed many people. (2) The problem solved by the principle is inherent in the way people conduct their lives and businesses, and thus the principle has been reinvented by different groups as they encounter the problem. (3) The impact of the principle was wide and deep and changed the practice of science, industry, or society.

Algorithmics (2 weeks)

An examination of great principles underlying all forms of computation. These principles address what machines (controlled by algorithms) can and cannot do and what we must do to circumvent inherent computational complexity. It also addresses misconceptions that many people in other fields have about algorithms and programming, including limitations of the metaphor of information. Statements of principle:

Simplicity and mechanicity: All algorithms are built from simple instructions that can be carried out by machines.

Translation: Programs are written in languages that can be automatically translated into codes executable by machines.

Universality: any computing machine can simulate any other.

The Church-Turing Thesis: any problem that can be solved by an effective procedure can be solved by a Turing Machine with appropriate program.

Uncomputability: Some problems cannot be solved by any algorithm.

Complexity: problems can be classified according to how many steps their algorithms take; many computable problems are intractable.

Heuristics: Some hard problems can be solved approximately by efficient approximations.

Self reproduction: some algorithms, when executed, produce exact copies of themselves.

Information: is information is a principle or an interpretation of certain actions of people or machines?

Distributed Systems (5 weeks)

An examination of great principles for designing reliable and dependable systems of clients and servers embedded in networks.

Modes of operation: stand-alone processing systems, multi-processor with shared memory, multi-processor with distributed memory, network clusters.

The Internet: survivable, packet switched, dynamically routed, protocols, WWW.

Naming: objects in large systems, hierarchical names, uniqueness in time and space, location transparency, dynamic binding, user choice.

Secure Communication: cryptographic protocols based on public-key cryptosystems and intractable problems gives methods for nearly unbreakable secrecy, signatures, and authentication.

Virtualization: four kinds, virtual memory, virtual machines, other simulations, information hiding and abstraction

Concurrent programming: processes, semaphores, mutual exclusion, determinacy, deadlocks, synchronization.

Transactions and data: atomic, two-phase commit, serializability, e-commerce.

Human Computer Interface: shell, GUI, translators, parsers, ergonomics.

Laws of system performance: basic laws, network model, computational algorithms, bottlenecks, measurement.

Cooperative Systems (3 weeks)

An examination of principles for systems that interact strongly with humans, help them coordination actions, and support their businesses and enterprises. Suggested topics:

Information systems

Agent-based computing

Speech recognition and understanding

Perception

Machine learning

Enterprise and E-Commerce

Contemporary Open Questions (2 weeks)

An examination of major contemporary open questions around which there is consensus of a big need for a solution but lack of consensus (and even controversy) about the nature of a solution. Examples from the current day:

Nomadic computing

Interaction models ("after the desktop")

Alternative methods of computing ("after Moore's law for silicon)

Oxygen architecture (MIT paradigm of design)

Bionic body parts

Books

Alan Biermann. *Great Ideas in Computer Science*. MIT Press, 1990.

James Burke. *The Day the Universe Changed*. Little Brown. 1995.

Richard Feynman. *Lectures in Computing*. Perseus. 1996.

David Harel. 2000. Computers, Ltd.: What Computers Really Can't Do. Oxford University Press.

Robert Hazen and James Trefil. *Science Matters*. Anchor. 1991

Danny Hillis. *The Pattern on the Stone*. Perseus. 1999.

James Trefil, Robert Hazen, Anthony Gaudin. *The Sciences: An Integrated Approach*. Wiley. 1999.

Grading Approach

Breadth requirement (60%, individual effort): students understand at least six of the great principles well; they demonstrate their understanding by providing written solutions to a set of Great Design Problems handed out at the beginning of the semester. Students are encouraged to submit drafts of their solutions for instructor feedback.

Depth requirement (40%, team effort): student teams complete a project of researching a great principle to document its history, its recurrences, its impact, and its durability. The project aims to assure that students have learned to be the kind of observer outlined in the first lecture. All students must participate equally in the team. Each team will submit a report give a public presentation of its system.

Teaching Philosophy

Class sessions can be a combination of moderately short presentations and open discussion about the history and implications of the principles presented. Student involvement is important in achieving a view of the whole.

Prior to classes, and in addition to classes, students can be asked to read from source books and other materials. They can be asked to write down questions they have from their readings and points on which they seek elaboration or clarification.

Students can be asked to write short essays on how a particular principle followed the pattern to being considered great. This develops their skills as observers of these phenomena.

Students can be asked to make their preliminary drafts for the Great Design Problems without consulting texts or references beyond their current knowledge. This encourages them to “invent it for themselves,” deepening their knowledge considerably.

In their team projects, students can be asked to contribute equally and various safeguards will be built into the operating rules for teams to assure this. All participate equally in the presentations of their projects. Project presentations give them practice in presentations and in responding to questions on the spot.