Computer Literacy for Undergraduates: A Document-Engineering Framework

Peter Wegner, pw@cs.brown.edu

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-94-21
Draft, June 1, 1994

Computer Literacy for Undergraduates:
A Document-Engineering Framework
Peter Wegner, Brown University, Draft June 1 1994

Table of Contents

Supplementary materials (not a part of this paper):
Assignment specifications and examples of completed assignments (400+ pages)
Overheads for lectures (400+ foils)

Computer Literacy for Undergraduates: A Document-Engineering Framework

1. Overview

“Concepts and Challenges of Computer Science” (CS2) is a course for undergraduates with no previous experience of computers that focuses on document browsing, creation, and management. It aims to elevate the collection of practical techniques associated with word processing, spreadsheets, hypertext, networks, and multimedia into a systematic discipline of document engineering. There are many deep analogies between document and software engineering, since large documents are like large programs in their life-cycle and linking structure. CS2 focuses on properties that are common to text, spreadsheets, programs, and multimedia documents. Programs are viewed as an important but not dominant class of documents that support animation, interaction and other forms of dynamic document behavior. Because software engineering for software documents has been extensively studied in computer science, its techniques can provide a framework for the broader discipline of document engineering.

The lectures of CS2 focus on computer literacy and fundamental concepts, while its assignments develop document management and technical writing skills. Its lecture structure is as follows:

Part I: Concepts
1. Perspectives on science and computing
2. Creating Macintosh and Hypercard documents
3. Computer hardware and software
4. Word processing, spreadsheets, databases, and networks

Midterm exam: Computer literacy test on basic computer concepts

Part II: Challenges
5. Artificial intelligence
6. Programming in Hypertalk
7. Social impact (economics, ethics, law,. security, education, workplace)
8. Large document structure (design, interaction, animation, document management)

Final exam: Material after the midterm exam

The lectures are coordinated with eight assignments and a final project:
1. Macpaint competition
2. Resume in Hypercard
3. How computers execute programs (in 12 Hypercards)
4. Spreadsheet assignment in Excel
5. Network treasure hunt and MS Word essay on networking
6. Hypertext on "Can Machines Think?"
7. Hypertalk programming assignment (cash register interface)
8. Essay in MS Word on "The Social Impact of Computing"

The final project, which takes a full month and includes approval of a design before proceeding with the implementation, requires creation of a substantial Hypercard, Mosaic, or Excel document. The final project alternatives in 1994 included:

Hypercard document:
  Education/information document (history, literature, sports, hobby, art, music)
  Adventure game (cave exploring, Brown campus, United States, Startrek)
  Hypertalk on-line user's guide (graduated examples, linked glossary, animated case study)
  Independent project (one that does not fit the above categories)
Mosaic document (locally created document linked to remote net documents)
Linked spreadsheets for business management (cash register, inventory, suppliers)

Of the 85 students in 1994, about 50 selected one of the Hypercard options, about 30 selected the spreadsheet option, and four selected Mosaic. The ten undergraduate teaching assistants worked with about ten students each helping them implement their creative ideas (see section 4).

The assignments, final project, and exams contributed to the final grade as follows:
  assignments: 50%
  final project: 25%
  exams: 25% (midterm 10%, final exam 15%)

CS2 provides a unifying conceptual framework for learning about word processing, spreadsheets, hypertext, networking, and multimedia. It enhances technical writing and presentation skills as well as document management skills, so that students taking this course in their freshman year can effectively use computers to enrich learning in later courses. It teaches computer literacy not only in the sense of "familiarity with computer terminology" but also in the sense of "ability to write clear (literate) computer documents". The course focuses on computer literacy in the first sense before the midterm, while the second part of the course deals with computer literacy in the second sense. The goals of CS2 are similar to those of Knuth's literate programming (Computer Journal 1984), but for documents rather than programs. We believe that five years from now first courses in computing will be closer in content to CS2 than to current introductory programming courses, even for computer science majors.

2. Assignments

The assignments determine the anatomy of the course, while the lectures determine its physiology. Assignments were chosen to build on one another in developing computer literacy, technical writing, and document management skills. The first two are pass/fail warm-up assignments, while the remaining assignments respectively provide hands-on understanding of hardware/software, spreadsheets, networks, artificial intelligence, programming, and social impact. We describe the course first from the perspective of its assignments and then from the complementary perspective of its lectures.

2.1 Assignment #1: Macpaint Competition

Drawing a picture provides immediate hands-on feedback, enabling students to use computers creatively after a single lecture on menus and Macpaint tools. It dispels fear of computers by an open-ended application that encourages ingenuity and imagination. Prizes were awarded in four categories: best artistic, best technical, funniest, and most original.

The class produced some excellent drawings of faces, sailboats, still life, landscapes, and airplanes with both artistic and technical merit. The prizes in each category were publicly exhibited and are shown in appendix 1. First-prize winners were treated to "coffee with the professor" at the coffee shop used later as the basis for the Hypertalk programming assignment and designated as the official coffee shop of CS2.

2.2 Assignment #2: Resume in Hypercard

The resume is assigned after two lectures on Hypercard, the first dealing with browsing and the second with authoring and linking. It further overcomes the fear of computing by encouraging students to talk about themselves in a structured Hypercard format. Students are required to create specific fields on each card of their resume and buttons that link the cards in a circular list. The structured resume has precisely four Hypercards:

Card #1: Name, address, optional photo
Card #2: Education, interests
Card #3: Goals in course, long-term goals
Card #4: Course topics of greatest interest, attitude to computers

Having learned Macintosh menus and Hypercard stacks in the context of nontechnical drawing and resume applications, students are now ready to apply these techniques to technical topics.

2.3 Assignment #3: Hypercard Stack on How Computers Execute Programs

The "How Computers Execute Programs" assignment requires students to develop an explanatory Hypertext on how computers work that discusses computer architecture, information representation, program representation, compiling, and program execution. It is preceded by lectures on hardware and software and by readings from the textbook, and is given two weeks into the course after the following lectures:

1. Introductory lecture
2. Macintosh menus and Macpaint tools: Macpaint competition warm-up assignment
3. Browsing in Hypercard: introducing stacks, cards, buttons, fields, backgrounds, the home card
4. Authoring and linking in Hypercard: resume warm-up assignment
5. Introduction to hardware: CPU, processor, binary and decimal numbers, logic gates, machine language
6. Introduction to software: Basic, Pascal, Fortran, Cobol, the idea of compiling: assignment #3

Students are asked to create a 12-Hypercard stack that starts with low-level hardware and information representation concepts and works up to high-level languages, interfaces, and hypertexts:

1. Table of contents (these 12 card titles)
2. Goals and audience (next year's students in this class)
3. Physical structure of computers (memory, cpu, input/output)
4. Logical structure of computers (programs, data, execution)
5. Representation of information (binary, decimal, characters)
6. Machine language programming (input two numbers, multiply them, output result)
7. Pascal programming (declare, read, multiply, print)
8. Compiling source code to machine language (language translation)
9. The Macintosh interface (menu, mouse, tools, interaction)
10. Hypercard (stacks, cards, buttons, fields, scripts)
11. Conclusion (symbol transformation, document management)
12. Epilogue (good and bad points of this assignment)

The problem specification describes the content and level of detail expected on each card. Almost all students showed substantial understanding of the subject matter in their assignments, suggesting that active learning by writing and explaining is a good way to internalize these concepts. Because the sequence of topics and the content of each card is specified, students can do the assignment by simply filling in blanks (the prose for each card). Most students spent between five and eight hours on this one-week assignment. The following epilog card illustrates student reaction to this assignment:

I had a great time with this assignment. I didn't intend to become involved with it, but once I was shown scripting, backgrounds, and effects I couldn't help but use them. They're so cool. I learned a lot about not only how to manipulate Hypercard in relation to other programs on the Macintosh, but also how to navigate better in general. The little short cuts and tricks are really neat. The only bad thing I can say about this assignment was that it was slightly long and tedious. I had an easier time than I expected because a TA showed me how to put buttons in the background and script them. The assignment was a lot of fun; just don't ever make us do anything this long again.

The assignment was considered tedious because all 12 cards had to be created and linked. Next year we will provide an initial stack skeleton with a table of contents and cards with some text, pictures, or programs already given. Providing a partial document is the document-engineering equivalent of asking students to augment a partial program. The assignment has both conceptual and technical goals:

1. Integrates understanding of different aspects of the program execution process (conceptual).
2. Consolidates knowledge of authoring in Hypercard, building on the resume assignment (technical).
3. Contributes to technical writing skills by helping students to organize and express their thoughts.
4. Introduces nonlinear Hypertext by requiring two-way links of the table of contents to each card.

2.4 Assignment #4: Cash Register Spreadsheet

Spreadsheets are specialized documents that display and manipulate tables of numerical, textual, and graphical data. The spreadsheet assignment, given at the end of the third week after lectures on and demonstration of Excel, specifies the sequence of steps required to create a simple cash register spreadsheet:

1. Enter text for title, row and column headings
2. Enter the formulae
3. Enter numbers to test the formulae
4. Format the spreadsheet for neat output
5. Print preview and print the spreadsheet
6. Optionally create a bar chart (column chart) of sales of each item

The detailed Excel operations for each task are given in the assignment, which may be viewed as a drill and practice exercise that complements concepts presented in the lectures. A more extensive spreadsheet program with linked spreadsheets to handle a business like a coffee shop was offered as a final project:

supplier information spreadsheet: name, address, items supplied, unit cost, delivery schedule
item information spreadsheet: item name, supplier cost, customer price, quantity sold, profit
cash register spreadsheet: list of items for sale, items sold, invoices for each customer
inventory: item name, number bought, sold, in stock
operating costs: electricity, rent, salaries, etc.
profit loss report: profits, taxes, audited accounts

Though producing this complete set of spreadsheets is too ambitious an assignment at this early stage of the course, we hope to expand the lecture material on spreadsheets next year and to replace this assignment with a more challenging linked cash register and inventory spreadsheet assignment. The spreadsheet assignment has the following pedagogic goals:

1. Describes the structure of documents for modeling accounts and businesses.
2. Introduces modes of editing, manipulating, and formatting for tables (arrays) of numbers.

2.5 Assignment #5: Network Treasure Hunt and Essay

The treasure hunt required the use of e-mail, access to newsgroups and on-line databases, and retrieval of remote network information. Items to be found in the treasure hunt are:

1. The two public telephone numbers of Brown's president listed in Brown's electronic address book
2. Sending e-mail to James Bond who replies with a secret password
3. Using the Gopher information-retrieval program Sextant to find information about a published paper
4. Using Newswatcher to browse through newsgroups and pick any three of personal interest
5. Using a file transfer protocol (ftp) to play music from an audio file at the University of Michigan

James Bond turned out to be a very effective secret agent, detecting students who handed in passwords they had "borrowed" from other students and gently reminding owners of duplicated passwords that it was wrong to give secret passwords away.

The main part of this assignment was to write a four-page essay in MS Word on the structure of the Internet describing e-mail, ftp, and experiences during the treasure hunt. This assignment generated considerable enthusiasm and its simple essay format was a welcome respite from the rigors of Hypercard. This assignment serves multiple pedagogic purposes:

1. The treasure hunt involves network "surfing", while the essay describes the principles of networking.
2. The essay requires word processing, one of the most important basic tools of computer users.
3. Technical writing skills are required to present this technical topic clearly.

As in the other assignments, the best submissions were remarkably good, and almost all students demonstrated substantial understanding of the subject matter. These five assignments, given during the first five weeks of the course, can be viewed as "computer-literacy" assignments that provide hands-on reinforcement in learning the basic terminology that determines computer literacy.

2.6 Assignment #6: Hypertext on Can Machines Think?

This assignment examines the relation between human and computer intelligence, exploring Turing's arguments for the position that machines can think as well as counter arguments by Searle that machines have no inner understanding and by Penrose that thinking is inherently more powerful than computation. It brings together fundamental philosophical and technical issues that lie at the heart of computer science. It is preceded by two lectures and a debate between the professor and the TAs on "Can Machines Think?".

Lecture 1: Behaviorist methodology of the Turing Test, Searle's and Penrose's objections to the test.
Lecture 2: Eliza's pattern-matching technique of simulating intelligence, its implementation in Lisp, discussion of whether such "echo intelligence" can be viewed as thinking.
Debate: Yes position argued by the professor, no position argued by four TAs with assigned roles. Each TA argued a particular position and prepard three foils rehearsed with the professor prior to the debate.

The following reference articles, discussed in the lectures, were made available to students:

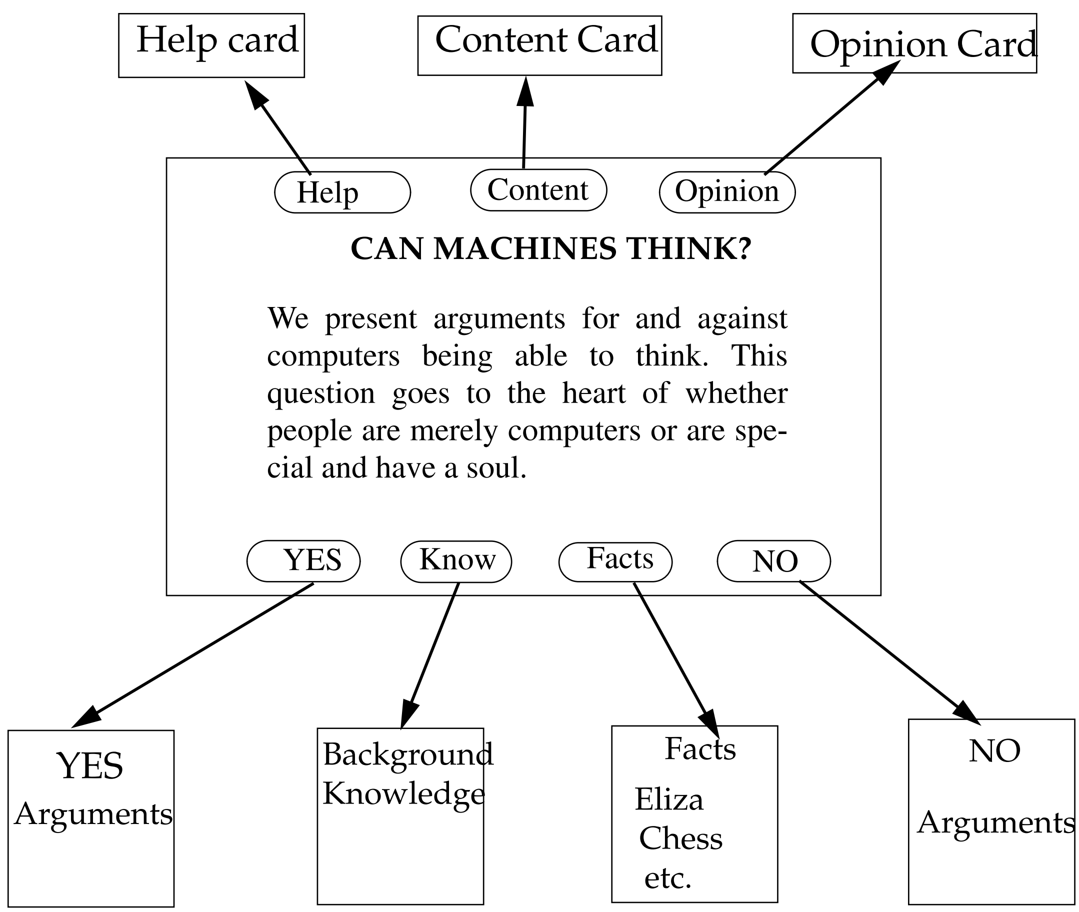
1. Turing's original paper "Computing Machinery and Intelligence" that proposes the Turing Test
2. Searle's paper "Minds, Brains, and Programs" arguing that computers have no inner understanding
3. Penrose's precis of "The Emperor's New Mind" arguing that computers are less powerful than humans
4. Isaac Asimov's story "The Bicentennial Man", describing a robot whose behavior becomes human

The assignment's high-level structure was specified by a "home card" with seven buttons (links):

1. A help button linking to a card to help you find your way around
2. A content button linked to a card with a detailed table of contents
3. An opinion card describing the student's personal opinion (with justifications)

Links to four sets of cards describing background, facts, yes, and no arguments

4. Background knowledge cards including a discussion of why the problem is interesting and what we mean by "machines", "think", "can", etc.
5. Facts cards that describe the Turing test, Searle's Chinese room experiment, Eliza, and other relevant facts independently of their interpretation as "yes" and "no" arguments
6. "Yes" arguments for the position that machines can think, linked to facts and background cards.
7. "No" arguments for the position that machines cannot think, linked to facts and background cards.



The detailed specification, combined with the lectures and reference materials, permitted students to develop uniformly informed and high-quality discussions of this topic. They had to grapple with questions like “what is thinking”, “what is intelligence”, “what is a machine” in the background section, understand Turing’s notion of computability and his claim that machines and people have equivalent computing power, and examine how behaviorist, rationalist, and empiricist philosophies deal with this issue. Next year we will simplify this assignment by giving students a skeleton stack with the home card and some background cards.

This assignment realizes the following pedagogic goals:

1. It provides a framework for arguments on a subject that engages all students in the course.
2. It applies Hypercard to a topic whose “yes” and “no” arguments have an inherently nonlinear structure.
3. It exercises technical writing skills on a topic that spans computing and the humanities.
4. It demonstrates the use of computers to augment our intellectual capacity by organizing ideas in ways that would otherwise be impossible.

2.7 Assignment #7: Cash-Register Interface Programming Assignment

This assignment involves design of a cash-register interface for a coffee shop consisting of five cards:

A main menu card with buttons for ordering tea, coffee, or muffins
Coffee, tea, and muffin cards that specify the type and cost of various brands of each of these items
An invoice card that computes and displays the total cost of an order

The main program (script) is an “on opencard” handler executed whenever the invoice card is opened. It transcribes orders of coffee, tea, and muffins from the three product cards onto the invoice card, listing for each item the quantity and total cost of the items purchased in a particular transaction. The script requires computing the total cost and the tax to complete the invoice specification. The program requires a double loop whose outer loop runs over the coffee, tea, and muffin cards and whose inner loop runs over the brands (of coffee, tea, muffins) on each of these cards. It has the following structure:

repeat with coffee, tea, muffin cards (for a sequence of card numbers)
repeat with products on each card (for a sequence of line numbers)
  if product has nonzero sales then enter product, quantity, cost on invoice card

This assignment involves both interface design and scripting (programming). It illustrates the role of scripting in realizing dynamic behavior in documents and that writing the program is a part of the larger problem of providing a useful service. Understanding the relation between the script and the multiple Hypercard interfaces makes this assignment non-trivial. It realizes the following pedagogic objectives:

1. It motivates scripting by a concrete cash register application for a local coffee shop.
2. It views programming as part of a larger process that includes interface design and everyday use.
3. It relates programming to business spreadsheet applications.

2.8 Assignment #8: The Social Impact of Computing

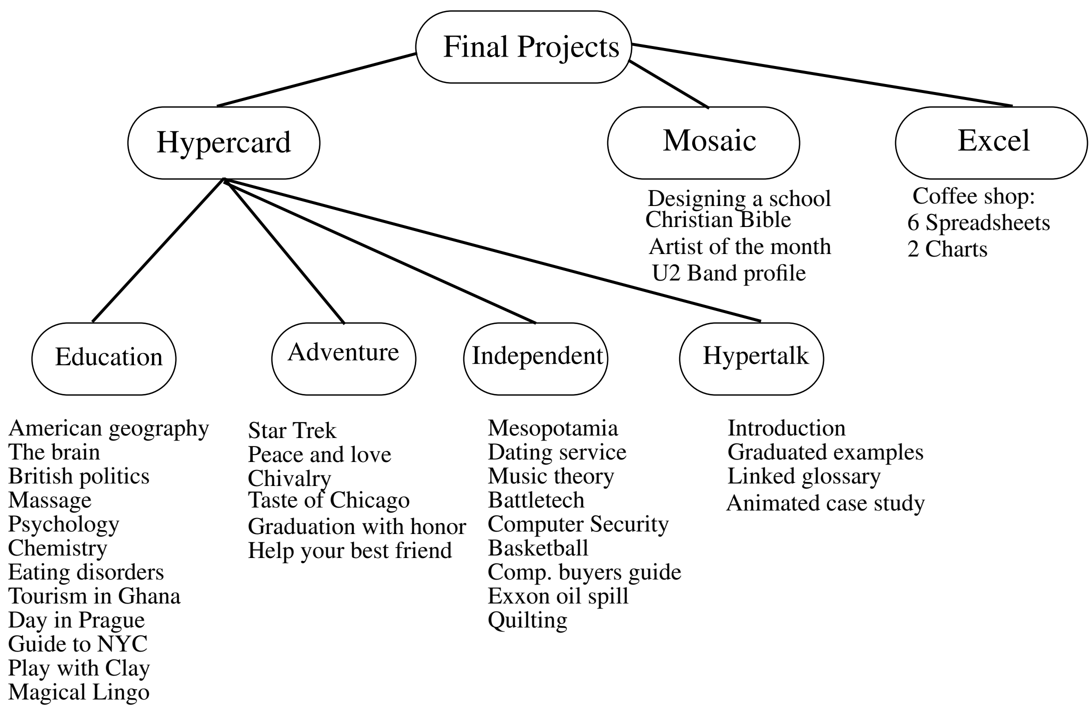
This assignment shifts the focus from the technical topic of programming to social and human issues in computing. Students are asked to write a four-page essay (on a word processor) that consists of an introduction followed by a discussion of the social impact of computing in one of the following four areas (corresponding to chapters in the textbook):

Chapter 11: Computers at Work
Chapter 12: Computers at School and at Home
Chapter 13: Computer Security and Risks
Chapter 14: Inventing the Future

The essays clearly benefited from the hands-on experience in the earlier part of the course, and were typically of greater than minimum length because students felt they had a lot to say. Students liked this assignment because it was a humanities-style essay rather than a hypertext. It provided an element of cultural literacy that complemented the course's emphasis on technical literacy.

2.9 Choice of Final Projects

The final project in CS2, like final projects in Brown's programming courses, allow students to work on a larger independently designed project. It provided a flavor of document engineering, just as final programming projects provide the flavor of software engineering. However, document engineering is of broader scope than software engineering and final project documents typically provide students with a greater sense of achievement than programs. Students can create documents that directly reflect their interests in music, sports, politics, or science. The figure below illustrates the range of topics chosen in 1994:



Final projects were organized so that students could build up momentum, starting from a high-level design idea developed during a one-week shopping period in which TAs made presentations of past assignments. Students then spent a further week developing a detailed design in consultation with TAs and the professor. The design was evaluated, to ensure that the project was both sufficiently substantive and not too ambitious. Students then had two weeks to implement their design.

The focus of the course was on Hypercard and the majority (60%) chose Hypercard as their framework for document development. A surprisingly large number (35%) chose spreadsheets, since they perceived them as being more directly related to applications they would encounter in real life. Only four students chose Mosaic, but they were among the most talented and hard-working, producing documents that were both polished and exportable. Educational and technical issues relating to final projects are discussed in greater detail in section 4 and in a separate document devoted entirely to final projects.

3. Description of Lectures

The early lectures are designed to provide basic computer literacy and a foundation for assignments, while later lectures examine the conceptual challenges and impact of computing. The lectures parallel the assignments, providing conceptual substance that transforms a collection of how-to techniques into a conceptually challenging introduction to information technology:

3.1. Perspectives on science and computing
3.2. The Macintosh and Hypercard
3.3. Hardware and software
3.4. Word processing, spreadsheets, and databases
3.5. Networks and what you can find there
3.6. Midterm exam focusing on computer literacy
3.7. Artificial intelligence
3.8. Programming in Hypertalk
3.9. Social impact of computing (economics, ethics, law, security, privacy)

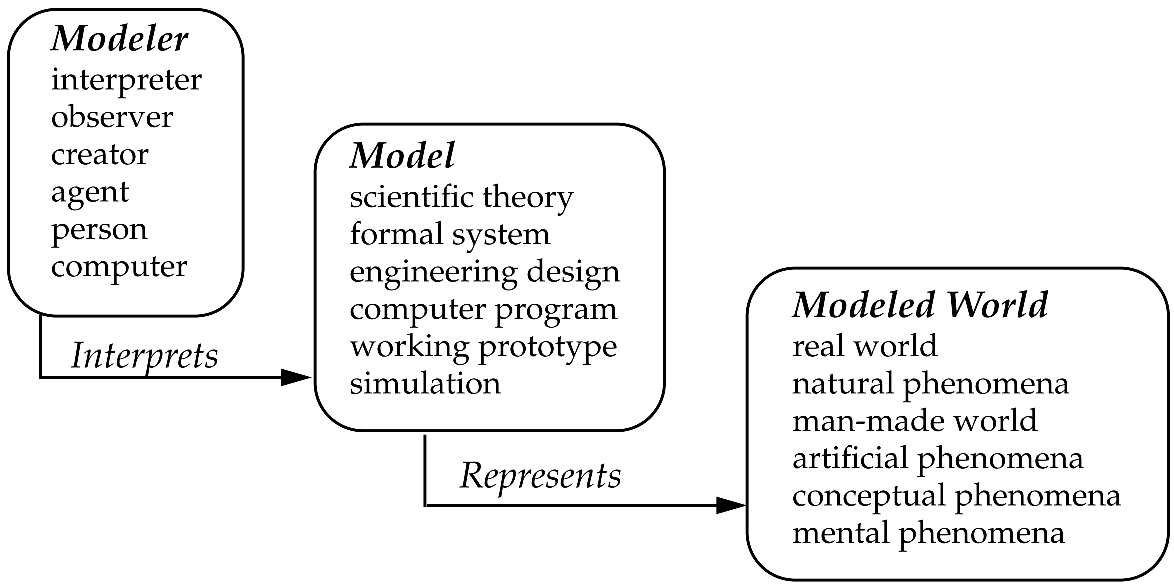
3.1 Perspectives on Science and Computing

The first lecture explores the relation between computing and established disciplines like physics, biology, engineering, mathematics, and philosophy. It examines the role of computer science as an engineering discipline and of programs and interfaces as models that represent and simulate application domains, Its broad introduction to scientific literacy provides a framework for computer literacy, while allowing "shoppers" to join the course in the second lecture without missing essential "how to" concepts.

The role of scientific method in physics and biology is examined, indicating that sciences build models that explain, predict, and control classes of phenomena. Sciences deal with abstractions that model objects as black boxes in terms of their observable behavior. *Natural sciences* such as physics and biology are concerned with properties of objects in the real world, while *artificial sciences* like engineering and computer science deal with the design and construction of man-made worlds. Natural science focuses on the analysis of phenomena in the real world, while artificial sciences deal with the synthesis of constructed artifacts.

Engineering deals with the same phenomena as physics, but from a different perspective. Traditional engineering supports the construction of physical artifacts, while software engineering supports the construction of conceptual software and information systems. Though software products are subject to broadly similar design and construction processes as physical products, they do not wear out, are easily copied, and are subject to different economies of scale and mass production.

Models are central to both the natural and the artificial sciences. However, software models of physical objects like airplanes are much more flexible than physical models; they can be more easily modified or embedded in larger systems like air-traffic control systems. *Models* in both the sciences and the humanities are *representations* of a *modeled world* that can be *interpreted* or *manipulated* for the purpose of *explaining* or *controlling* its phenomena. Modeled worlds can be real worlds of natural phenomena in the natural sciences or artificial worlds in the artificial sciences. Models can be scientific theories, formal systems, or computer programs. Representations of modeled worlds can interpreted by a human observer or a computational agent. Software models are generally interpreted by computing agents rather than human agents.



Mathematics and philosophy are also concerned with modeling, but their conceptual models have properties that need not correspond to the real world. Some conceptual models, like those of geometry, are designed to capture physical properties, but may also model unfamiliar (noneuclidean) properties. Plato argued that mathematical (ideal) objects are more real than physical objects because we can be more certain of their properties. Plato's parable of the cave asserts that we cannot perceive reality but merely its reflections (shadows) on the retina of our sense perceptions. The real world is an impenetrable black box.

Physics can talk only about the black-box behavior of physical objects and has no way of observing the reality inside black boxes. Computer science is concerned with the study of black-box behavior of hardware and software, in part to better understand how they function and in part to understand their internal structure so we can construct and modify their functionality. It provides a flexible general-purpose framework for modeling real, artificial, and conceptual worlds. It explores both models of computing and models of phenomena in other sciences.This lecture introduces the following basic vocabulary:

computer, science, physics, biology, natural science, artificial science, engineering, computer science, hardware, software, software engineering, black box, abstraction, model, modeled world, representation, interpretation, mathematics, conceptual sciences, reality of mathematical objects, Mandelbrot sets, Plato, Plato's cave, Platonic objects, computational objects, computers as black boxes, programs as black boxes, interfaces, algorithms, processes, memory

3.2 The Macintosh and Hypercard

In the second lecture we get down to the practical business of using computers, introducing the Macintosh Window, Icon, Mouse, Pointing (WIMP) computing style and the Macpaint tool box. Our goal is to provide a basis for immediate hands-on use as well as some basic vocabulary which is a part of our computer literacy checklist that is tested on the midterm exam:

Macintosh, screen, keyboard, mouse, button, pointer, cursor, floppy disk, hard disk, icon, folder, click, double click, menu, pull-down menu, window, application, Macpaint, MS Word, Excel, desktop, desktop metaphor, pointing, pressing, dragging, file menu, new, open, save, print, edit menu, copy, cut, paste, window operation, close box, zoom box, title bar, scroll bar, size box, balloon help, switching between applications, Macpaint tool box, drawing tools, pencil, painting tools, paintbrush, selection tools, lasso, drawing faces, drawing houses

The Macpaint assignment provides hand-on experience in the use of menu and toolbox facilities for drawing. A demo that features drawing, painting, selection, copying, cutting, and pasting is given during the lecture and repeated in evening laboratory sessions run by teaching assistants. The Macintosh lecture is followed by two lectures on creating, editing, and linking Hypercard documents. The first of these introduces stacks, buttons, fields, and dialog boxes and demonstrates the browsing facilities provided by the home stack, while the second covers the creation of buttons and fields and links between cards that support browsing and navigation. These lectures provide a basis for the resume assignment and introduce the following vocabulary:

Hypercard, stack, cards, buttons, fields, home stack, user level, browsing, typing, painting, authoring, scripting, dialog boxes, navigation, background, object menu, creating fields and buttons, editing fields and buttons, tools menu, browsing tool, button tool, field tool, button dialog box, show name, select icons, effects, script, link to, style, go menu, recent, arrow keys, linking to new card, create two-way link, simple scripting, on mouseup handler, go to script, message box, hide menu bar, show menu bar

Linking is a relatively advanced topic in traditional courses on programming taught towards the end of the course. However, linking of a collection of Hypercard documents is considered as fundamental as the techniques for specifying their content and is therefore taught very early. The resume assignment provides hands-on practice of Hypercard authoring and linking.

3.3 Hardware and Software

Hardware and software concepts are introduced after students have mastered Macintosh and Hypercard basics. Lectures are supplemented by textbook chapters and further reinforced by the assignment. The hardware lecture covers concepts of computer architecture, machine-language programs, binary, decimal, and ASCII information representation, and logic gates for adders:

hardware, software, Von Neumann computer, vacuum tubes, transistors, integrated circuits, mainframe, microprocessors, instructions, data, gates, central processing unit (CPU), arithmetic logical unit, instruction control unit, instruction counter, machine language, assembly language, operation code, address code, symbolic addresses, read and write instructions, load and store instructions, arithmetic and logical instructions, transfer of control instructions, pascal program, compiler, source language, target language, translating source to target language, execution, debugging, information, binary digit (bit), byte, kilobyte, megabyte, number representation, ASCII, decimal addition, binary addition, logic gates, half adder, adder, game of Nim

The single lecture on hardware is followed by a single lecture on software in which basic programming concepts like statements, conditions, loops, languages, compilers, and operating systems are introduced:

programs, recipes, machine language, procedure-oriented language, software engineering, structured programming, object-oriented languages, graphical user interfaces, Algol 60, Pascal, simple program, expressions, statements, declarations, loops, sum, mean, complete program, subprogram, Fortran, basic, *C, common applications, packages, software licenses, look and feel, system software, syntax, semantics, pragmatics, lexical analysis, semantic analysis, code generation, operating systems, data structures, arrays, records, trees, lists, examples of databases, interfaces, operating system shells, UNIX, MS DOS, graphical user interfaces, WIMP, networks, local area networks, wide area networks, e-mail*

The number of concepts in these lectures is quite large, but supplementary reading allowed this basic material to be covered quickly. Since these concepts are not difficult, active learning by writing about them seems preferable to risking boredom by a longer presentation. The lecture and textbook material was immediately reinforced by the assignment on "How Computers Execute Programs".

The layered "onion" model views computer systems as layers with hardware at the inner level and system software, application software, and end-user interfaces at progressively outer levels. The lectures examined the relation between low-level and high-level descriptions. The progression from logic gates, bit representations, and architectures for hardware is paralleled by the relation between machine language, high-level languages, and applications for software. The evolution of hardware from vacuum tubes in the 1950s through transistors, integrated circuits, and personal computers to networks in the 1990s was paralleled by the evolution of software from machine language in the 1950s through procedure-oriented languages, software engineering, and object-oriented languages to graphical user interfaces in the 1990s.

The structure of the CPU and of instruction execution was discussed and the mapping of a simple Pascal program onto an equivalent machine-language program was examined. Simple equivalent programs in Basic, C, Fortran, and some other languages were shown to give the flavor of traditional programming languages. The notion of system software was illustrated by compilers and operating systems and the role of software in supporting graphical user interfaces like that of the Macintosh was discussed.

3.4 Word Processing, Spreadsheets, and Databases

The next three lectures cover word processing, spreadsheets, and databases. The word-processing lecture briefly discusses the evolution of language, writing, printing, typing, word processing, and electronic publishing. It examines the structure of documents, and the relation between text editing with format commands and WYSIWYG editing. The menu, ribbon, and ruler operations of MS Word are described, different kinds of fonts and styles of text are presented, and print preview is discussed. To provide perspective on other word-processing systems, the structure of documents in framemaker is briefly described. Creation of small one-off MS Word documents is contrasted with maintenance of large long-lived documents in libraries and document archives.

Certain text-editing operations like selection, dragging, and cut-and-paste apply generally to editing visual and audio as well as textual documents. Cut-and-paste and dragging are manually performed assignment operations that, like assignment in programming languages, apply to all types of documents. These operations are first introduced as text operations of MS Word, but the idea that selection is a first-class operation applying to all document types is introduced at this early stage. Rudimentary multimedia document editing is illustrated by importing a Macpaint picture into an MS Word document.

Spreadsheets are an important class of documents with a very different structure from word processors, consisting of tables (arrays) of numbers arising in accounting and business applications. The spreadsheet lecture analyzed the structure of a particular cash-register spreadsheet, demonstrating how to create it by entering the text headings, formulae, and numbers into a blank Excel spreadsheet form. A grade spreadsheet was introduced and used to illustrate sorting on selected data like student names or final grades.

The use of spreadsheets to answer what-if questions like “What if I had got ten more points on a particular assignment?” was discussed. A cash-register spreadsheet linked to an inventory spreadsheet was introduced, and the idea of linking cash-register sales to inventory to adjust the inventory automatically on each sale was discussed. It was shown that the links among columns of a single spreadsheet established by formulae are similar to the links among columns of different linked spreadsheets: in both cases a real or what-if change in one part of a spreadsheet system automatically updates dependent (linked) quantities. The similarities and differences of links in spreadsheets and links in hypertexts were briefly discussed in an attempt to develop a general notion of linking and sharing in computer documents.

Spreadsheets provide a starting point for the discussion of databases. We review database applications such as airline reservation, credit cards, grades, social security, and banking that are central to the functioning of modern society. The tabular structure of databases is examined, and query languages for selecting information in databases are discussed. The use of graphical user interfaces as front ends to databases is discussed, and some examples are given. Spreadsheets are shown to be a special kind of database that provides facilities for tracking and linking dynamically changing tables of data.

3.5 Networks: Browsing, Communication, and Document Acquisition

Computer networks connect individual users together into a global community (global village) through information highways. Users are part of a local area network (LAN) that contains local information about members of the Brown community and about the academic and social infrastructure of the university. Users are connected to geographically distant computers through a wide-area net (WAN) that supports electronic mail (e-mail) and the retrieval of distant information through file transfer protocols (ftp).

The lectures on networks cover the history and evolution of networks and the Internet, introducing electronic mail, finding information in local files, and getting files from remote locations by remote login and file transfer protocols (ftp). The lectures were supplemented by detailed information on how to use Brown’s electronic address book (EAB) to look up information on Brown’s employees, instructions for setting up and using e-mail, using local information retrieval facilities to look up course schedules, news, and weather, accessing newsgroups and bulletin boards available at Brown, and using a file transfer protocol to retrieve and acquire nonlocal information through anonymous ftp. The Mosaic system for creating, retrieving, and accessing multimedia documents was discussed but not actively used, since it was not at that time accessible on the student Macintosh network. Techniques of multimedia document preparation such as the scanner, audio input, and visual editing tools like Adobe Photoshop were briefly mentioned though they were not discussed in detail till students started on their final projects.

The network part of this course included a skit written by the TAs featured a professor plotting to take over the world through the Internet. The skit opens with the professor asking secret agent 002 (the course was numbered CS002) to gather information on the net. The main part of the skit involves agent 002 asking various TAs to perform Internet assignments to gather the information, and ends by agent 002 discovering the plot and unmasking the professor.

3.6 Midterm Exam (Computer Literacy)

The midterm exam is a computer literacy test, where literacy is defined in terms of a vocabulary of about 300 technical terms covered in lectures and assignments, including those in italics above. Defining computer literacy by a list of words provides an accessible, sharable literacy standard as well as a framework for students to internalize the concepts learned during the first five weeks of the course. The vocabulary list is distributed and discussed prior to the exam and students have an opportunity to ask about individual words, both during the review lecture for the exam and in an evening review session run by the TAs. The questions in the 1994 exam had the following form:

What are the principal similarities and differences between the following pairs of terms?
hardware and software
Macpaint and Hypercard
hard drive and floppy disk
arithmetic unit and control unit
declarations and assignment statements
if-then and loop statements
electronic mail and file transfer (ftp)

This kind of question requires relations among concepts to be appreciated and provides n-squared possible relations among a set of n concepts (90,000 relations for 300 terms). We tell students that the questions will have this form, and even give them last year's exam. Making up an exam of this kind is easy. Students can use up to 4 lines (50 to 100 words) to answer each part of each question. The exam lasts one hour and involves 15 to 20 such questions distributed over the subject areas covered in lectures and assignments.

This "vocabulary list" approach to defining computer literacy mirrors that of E. D. Hirsch to cultural literacy (Houghton Mifflin 1987) and seems to work quite well in providing a meaningful standard for computer literacy. The list used in CS2 is not definitive, and computer literacy is more volatile than cultural literacy, changing with technology. But it helped to provide a well-defined and well-understood core vocabulary (over 90% of the students scored at least 70% in the test).

3.7 Artificial Intelligence

Artificial intelligence is a conceptually challenging area of computer science that appeals directly to non-majors. We briefly review areas of artificial intelligence such as natural language understanding, expert systems, theorem proving, robotics, and perception. A simple Eliza script is presented and the pattern-matching methods of sustaining such a dialog are described in LISP, thereby providing students with a reading knowledge of LISP. The question "Can Machines Think?" is examined using seminal papers by Turing and Searle as well as a story by Isaac Asimov (The Bicentennial Man) featuring a robot who becomes human. The assignment required students to organize the arguments for and against "Can Machines Think?" in a Hypertext. The three lecture periods devoted to the "can machines think?" topic included a debate sandwiched between a concepts lecture and a LISP-based Eliza lecture:

1. Analysis of the Turing Test as an experimental criterion for testing whether machines can think and of the behaviorist (positivist) philosophy that underlies it. Presentation of Searle's Chinese-room experiment showing that competence does not imply understanding and of Penrose's view that the laws of physics and therefore the physiological laws governing human thinking are non-computable.
2. A debate of the professor versus the TAs, where the professor presented Turing's position, four TAs each presented specific counterarguments, and students from the class presented their arguments. The vote after the debate overwhelmingly supported the position that computers cannot think.
3. A detailed examination of the Eliza Doctor program and of how Eliza handles interactive conversation. Schematic LISP code was presented to show how Eliza operates by recognizing patterns and then transforming them. This lecture served to introduce the programming language LISP and the conceptual role of pattern matching in intelligence. Though Eliza reinforces the idea that computing without understanding is possible, the idea of emergent behavior in both humans and computer programs is introduced to show that Eliza can also be interpreted as supporting the position that machines can think, since the behavior of the

whole can be greater than the sum of its parts.

The use of "Can Machines Think?" as the driving force for organizing lectures in this area narrowed our scope, but focused the attention of students on a set of conceptually intriguing topics, and caused them to internalize these ideas through the assignment. They had to express their opinion on topics such as behaviorism, robots, and computability and to describe the structure of programs like Eliza that simulated intelligence superficially without any real understanding. Here is a card from a student hypertext entitled: "What is thinking and why is it interesting to talk about whether machines can do it?"

Most people associate thinking with exercising mental faculties to form ideas, derive conclusions, and follow a chain of reasoning. I don't believe anyone will doubt that the human brain is capable of thinking. People are interested in the idea of whether machines can think because it in part defines who we are. For instance, by declaring that machines can think, do we relegate out status as superior beings with superior capabilities? When we agree that humans think, what exactly does this imply? For some it implies understanding and the ability to reason. If this is so, the next question is how does one test for this? Turing believed that "thinking" was testable by the Turing Test. To see exactly what this method was, push the "Turing" button. To see Searle's Chinese Room experiment, which is an objection to the Turing test, press the Searle button.

By organizing the rich body of ideas in this area into a Hypertext, students learned both substantive ideas of some great thinkers on the foundations of computing, and document management techniques for structuring a nontrivial corpus of concepts into a linked Hypertext document. The principle of using computer tools to express ideas about computers is familiar in programming, where programs that "understand" how programs are executed (interpreted) are written in the language itself. Here this idea is applied in a slightly different way to using computer tools in creating documents about topics in computing.

This assignment involved interdisciplinary concepts spanning computing, philosophy, and scientific method. Though it specified the structure of the Hypertext in considerable detail, students were responsible for organizing the required information into cards and for links among related cards. Links between facts and their yes and no interpretation and links from the opinion card to particular explanations and arguments were suggested. This assignment was the longest of the assignments (other than the final project), requiring between 25 and 35 Hypercards and taking between 8 and 12 hours, but it generated great enthusiasm. In next year's course we plan to simplify this assignment by providing students with an initial stack that contains the home card and some of the background and fact cards.

3.8 Programing in Hypertalk

Students learned how to read programs well before they learned how to write them. Programs were examined in earlier lectures on software, Hypercard, and Eliza, but the first programming (scripting) assignment was given late in the course, after the "Can Machines Think?" assignment. Scripting in Hypertalk was covered in two lectures. The first covered messages, handlers, basic commands, data structures consisting of fields, and conditional branching and loops, and a program for computing the sum and the average in Hypertalk. The second covered handlers, including the normal case of "on mouseup" handlers activated at the end of a mouse click, the effect of "on mousedown" and "mousestilldown" handlers, and handlers for "on opencard" and of programmer-defined functions. The message-handling "inheritance" hierarchy, which services handlers first for buttons and fields and then for cards, backgrounds, stacks, the home stack, and the Hypercard system, was illustrated by scripts for beeping.

In the second lecture the detailed syntax and semantics of basic commands were specified and methods of moving data between cards were explored. The “on opencard” handler was illustrated for moving data and performing actions when a card is opened. A program with a double loop was presented in which the outer loop was a sequence of fields and the inner loop was a sequence of lines in each field. The structure of this program was similar to that of the coffee-shop assignment for moving data from cards containing sales information to an invoice card and computing the total amount sold.

The programming assignment (to develop the interface and script for a coffee-shop cash register) was simple by the standards of traditional programming courses. It was motivated by a practical application and provided visual execution-time feedback. Hints and even pseudocode were given in the assignment. Even so, many students found this assignment difficult and needed extra help from TAs.

Random number and animation features of Hypertalk were presented and demonstrated so they could later be used in final projects. Random numbers were motivated by the outcome of fights in adventure games between the player and monsters encountered during an adventure. To illustrate the simplicity of the programs required for randomness and animation we give some Hypercard program fragments.

A random number in the range 0 to 100 can be generated by the expression “random of 100” and stored as the value of a variable, say x, as follows:

```
 put random of 100 into X  -- this puts a random number between 0 and 100 into X
```

To determine the outcome of a fight between you and a monster in an adventure game, we can generate two random numbers that represent playerpower and monsterpower and determine who wins the fight by comparing the two variables:

```
 put random of 100 into playerpower
 put random of 100 into monsterpower
   if playerpower > monsterpower then put 1 into playerwin else put 0 into playerwin
```

At a later point, we can use the value of the variable “playerwin” to output a message to the player:

```
 if playerwin then answer “the monster begs for mercy and gives you the magic lantern”
 else answer “the monster kills you and you wake up in bed in a cold sweat”
```

A given Hypercard can be animated by dynamically changing the location of fields and buttons on the card. The screen is represented as a two-dimensional 512* 342 grid of pixels. The origin (0,0) is in the upper left-hand corner and pixel coordinates increase downward and to the right.

The location of buttons can be accessed by the command “loc of button id”. The following command gets the first coordinate of the location of the button with id = 3 and puts it into the variable X:

```
 put item 1 of loc of button id 3 into X  -- get x-coordinate of button location and put in X
```

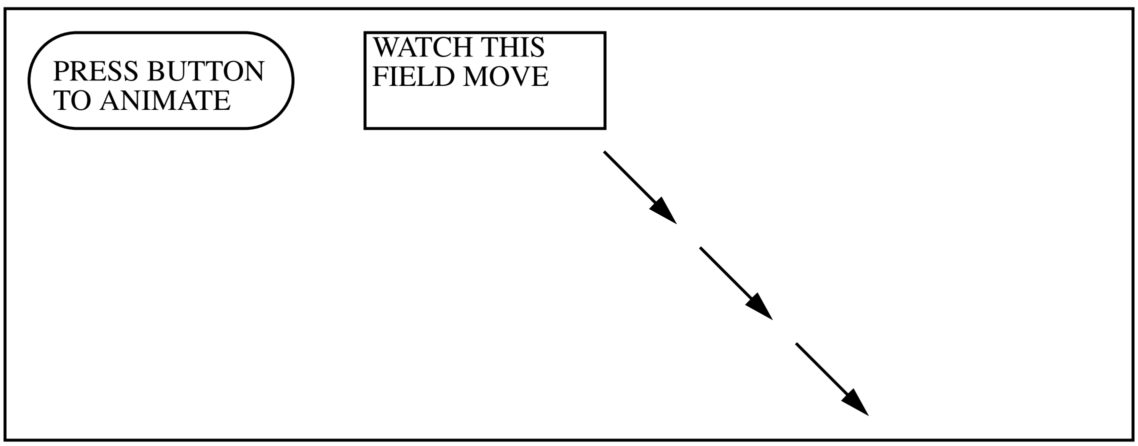
The following simple animation script moves the button with id 3 20 pixels downwards and to the right. It waits 30 ticks (half a second) between successive iterations. The total movement of 120 pixels is about 1/3 of the screen vertically and 1/4 of the screen horizontally:

```
on mouseup
 put 20 in dx  -- set the x increment to 20 pixels
 put 20 in dy  -- set the y increment

 put item 1 of loc of button id 3 into X  -- get the x-location of the button
 put item 2 of loc of button id 3 into Y  -- get the y-location of the button
```

repeat 6 -- repeat the loop 6 times
add dx to X
add dy to Y
set loc of button id 3 to X,Y -- set the new location of the button
wait 30 ticks -- wait for half a second
end repeat
end mouseup

If this script is in a button whose name is "ANIMATE" and the button with id 3 has a pretty picture, then the picture will move downward to the right six times whenever the ANIMATE button is clicked. To repeat this, drag the button back to approximately its original position and click again.



3.9 Social Impact of Computing

The social impact of computing was examined eight weeks into the course, after students had had intensive hands-on exposure to computers and were computer-literate. The course became less technical and more like a humanities or social science course. This part of the course included visiting lecturers about once a week on topics like robots, computer art, computer education, and computer graphics.

The first social impact lecture, on computer economics,. examined the transition from IBM's vertical integration approach that provided all computer services from one manufacturer to horizontal integration by unbundling computer services. Horizontal integration was illustrated with chip manufacturing by Intel, operating-system development by Microsoft, and application development by Lotus, all addressing different segments of the market. The economics of the Pentium versus Power PC chip, a hot topic at the time, was discussed and several current articles on this and other current topics were distributed.

The second social impact lecture on the ethics of computing used the ACM Code of Ethics(Communications of the ACM, Feb 1993); students presented case studies from that article on topics like intellectual property, privacy, confidentiality, and conflict of interest.

The lectures on computer security used material from the textbook supplemented by detailed analysis of how the Internet worm of November 1988 breached security using:
defect in the debug mode of sendmail
data overflow in the finger command
breaking passwords by guessing them

The legal implications of the indictment and sentencing of Robert Morris for this prank were discussed, including the fact that he received a significant, though relatively light, sentence. This section of the course also included a debate on the federally proposed legislation to require wire tapping, based on the article "To Tap or Not to Tap" in the February 1993 issue of Communications of the ACM. In arguing for and against federally mandated wiretapping, TAs were asked to focus on the following issues:

1. Required built-in listening devices (the clipper chip) would hamper innovation and competitiveness
2. The security and privacy of individuals would be jeopardized
3. The costs would be enormous and would exceed benefits
4. Criminals could circumvent these devices by private encryption of their messages
5. The rules for compliance are fuzzy

The assignment on social impact was an essay in MS Word on one of the following four topics, corresponding to chapters of the textbook:

Impact on the workplace
- changed organization and management practices
- changed work environment and production technology
- job satisfaction, social dynamics, working in the home
- impact of tools like word processors, networks, memory aids

Impact on education and the home
- more flexible ways of learning adapted to individual learning abilities
- multimedia impact on learning and leisure
- access to more information and to people at a distance

Security, privacy, and risks
- intellectual property and its protection
- safeguards against invasion of privacy
- moral and legal issues of prosecuting hackers and computer crime
- should court-ordered wiretapping be permitted?

Impact on the future
- hardware evolution in the next ten years
- software evolution: ubiquitous computing, intelligent agents, virtual reality
- evolution of quality of service: combining computers, telephones, television
- changes in our way of life: mental and physical effects

This assignment motivated students to organize for themselves the material on social impact presented in class and to go into detail on one specific area of social impact.

4. Final Projects and Final Exam

The final project is a capstone learning experience in which students do a project of their choosing using tools learned in previous assignments. The lectures during the one-month final project period cover project-related topics, including principles of project design, analysis of case studies of adventure games and educational hypertexts, and discussion of multimedia tools like audio-help, the scanner, and Adobe Photoshop. Students commit themselves to a final project during the first week, preparing a detailed design in the second week and then having two weeks to implement their design. TAs in the course work individually with up to ten students each on final projects. The four-week final project period is organized so that students must meet checkable goals along the way:

First week: a high-level (one-paragraph) description, meet with TA to discuss detailed design
Second week: detailed design due (20% of grade), reviewed by TA and professor
 second meeting of student and TA to review detailed design
Third and fourth weeks: implementation with technical help from TAs
 TA sets an implementation milestone to be checked at end of third week

In addition to these meetings there are help sessions on the use of tools for audio and visual effects and of tools for animation using Hypertalk. Group meetings encourage interchange of ideas among students in each project category. Students are encouraged to use the help of TAs to achieve interesting effects. Both students and TAs found this form of interactive creativity one of the most rewarding parts of the course.

The final exam provided an incentive to review the topics of the course. It covered Hypertalk programming as well as topics like economics, intellectual property, security, and document design from a conceptual point of view. It complemented the practical and narrowly focused final project, maintaining the balance between conceptual substance and hands-on experience that was one of the course's primary goals.

4.1 Document Engineering

Document engineering deals with the management of large, long-lived documents over their life cycle. For example, a repository for journals of a professional society must serve the needs of authors, referees, editors, and copy editors during the manuscript preparation process and the needs of readers, accountants, reference librarians, and others with an interest in the finished product. Other documents, like a company's financial records or a multimedia film library, have different requirements for document design, enhancement, archiving, and delivery. Military systems like battleships may have millions of lines of documentation prepared by hundreds of contractors and have adopted SGML (Standard Generalized Markup Language) as a document interchange language in their "Computer-Aided Logistics and Support" (CALS) system. SGML marks up documents by tags that identify their components. It supports text interchange (interoperability), providing a portable structure specification for document exchange and composition.

Documents have an inner *structure* among their parts, a *link structure* that connects them to other documents, and access restrictions that may prevent unauthorized users from modifying, accessing, or copying them. Hypercard is a single-user system that does not scale to multiuser systems. It has no "find" mechanism for finding words or patterns in large documents. Access to Hypercard stacks is controlled by the operating system. Multiuser systems like Mosaic require an extra layer of description specified in markup languages like SGML or HTML that support scalability, composability, interoperability, and other mechanisms of document engineering. Document engineering provides a natural conceptual framework for CS2 and the 1995 version of this course will examine SGML, Mosaic, and document ttype definitions (DTDs) from the point of view of document engineering and include them at an earlier point in the course.

In designing a document we must ask questions like "Who are its clients?", "How will it be used", "What is its lifetime?", "Who is entitled to access it?". The Hypercard documents of CS2 are not archival but are nevertheless sufficiently complex in structure so that document engineering questions of modular design and quality of presentation are important. Students must be comfortable with links among components of a single document both for Hypercard and spreadsheet applications (where links represent constraints rather than merely connections for purposes of browsing). Links from a Mosaic document to a remote repository are in some respects similar to Hypercard links and in other respects different. The primary goal of final projects is quality of content, but awareness of the larger context of document engineering for multiuser distributed documents provides a framework for good document design.

4.2 Hypercard-Based Final Projects

Students could choose Hypercard-based projects from four categories.
1. Education/information projects that support the learning process for academic or speciality topics or simply present information in an interesting way.
2. Adventure projects that involve a protagonist with a goal who must use intelligence and skill to realize the goal in a make believe world like cave exploring, spaceships, or the Brown campus.
3. On-line Hypertalk guide that involves development of an animated education document for one of the hardest topics in the course.
4. Independent Hypercard projects that did not fit in to any of the above categories.

The high-level descriptions below for each of the four classes of Hypercard projects are similar to those actually provided to students. The high-level description due at the end of the first week should give TAs sufficient information to advise on the feasibility of the project, and should provide evidence that the student has devoted some thought to the topic.

Example education scenario: A Journey through time
Brief description: "A Journey Through Time" is an educational program about world history. The user is taken on a "journey through time" with the ultimate goal of locating a chalice hidden in a certain historic period. The user is aided in her search by clues that she is given along the way. Her mastery of the historical period is tested, and if she performs well, she gets clues pointing to the chalice's hiding place.

Example adventure scenario: Quest for Vartan Gregorian
Brief description: This adventure game is a quest for an interview with President Vartan Gregorian. The catch is that an interview requires a contribution to Brown's Capital Campaign, and you have to find or earn money before an interview will be granted. You will have a chance to find, earn, or lose money as you roam around your favorite campus haunts.

On-line Hypertalk guide: no choice of high-level design, four main sections
What is Hypertalk? (relation to Hypercard, goals, motivation, audience)
Graduated examples (from simple to more complex ones)
Glossary (linked to other sections)
Coffee-shop case study (explanation and animation)

Providing a more detailed level of design than in other Hypercard projectslet students focus on substantive Hypertalk questions. The detailed design for this project required specification of the sequence of graduated examples and a discussion of how the case study of the coffee-shop assignment could be made educationally meaningful. Developing the guide helped students attain a deeper understanding of Hypertalk and of programming in general. The best guides were quite professional in providing an understanding of the subject matter and we plan to use them next year in teaching Hypertalk more effectively.

Example independent project: Exxon Valdez oil spill
Brief description: Describe the initial accident and the process of environmental cleanup, the technological, legal, and social questions raised, the effect on wildlife, and other factors. Illustrate with pictures of the ship, the oil slick, the beaches, and other relevant pictures.

Before embarking on a detailed design, students were required to discuss their ideas in the first of two design-check meetings with their TA. The class lectures during this period were devoted to design from the perspective of software engineering and document engineering. The idea of a document life cycle involving design, integration, and enhancement was developed by analogy with the software engineering life cycle. Lectures on the process of design were supported by suggestions of specific topics for document development such as:

History: the American Revolution, the Vietnam war
Literature: Charles Dickens, African literature
Specific work: Hamlet (play), Bible (book), Casablanca (film)
Specific place: New York, Providence, Brown, your favorite art museum
Specific person: Thatcher, Gandhi, Castro, Friedan, Clinton
Arts and crafts: weaving, pottery, flamenco music

We suggested a document-engineering discipline for narrowing the focus of Hypercard projects, starting by identifying the goals and audience. Are the goals to acquire specific knowledge or skills, or simply to provide information and understanding of a topic? How much testing, if any, will be performed? Will the audience be school or college students, hobbyists, specialists, etc?

The mode of user interaction, and the multimedia facilities to be used should be specified. Is the document descriptive, interactive, a game, a simulation, an animation, etc.? Will Hypertext, graphics, scanned-in pictures, sound, video, etc. be used?

Finally the sources of information should be carefully specified Will the information be taken from the net, from books, in film libraries, etc.? Students are encouraged to work on a topic about which they already know a great deal so that they can focus on organizing the computer document rather than learning about a new subject area.

The questions above can be organized as a questionnaire that students fill in as part of the process of narrowing the focus of their project.

4.3 Design of An Adventure Project

“Design” can be defined as an outline showing the main features of something or as an underlying scheme that governs the functioning of a system. It is an integral part of the creative process in architectural design, dress design, or interior design. Software design is less tangible than design of physical products. But document and Hypertext design are less abstract than program design, though they may have hidden scripts that define their dynamic and interactive properties.

The goals of design are to describe the essence of a complex product or system for the benefit of one or more clients of the system. In creating a document, we are concerned primarily with the design from the point of view of the creator, though we are designing the system for a class of users and must therefore consider how the system will be used. The design document produced for the design review must specify both *how* the document will be produced and *what i*t will contain.

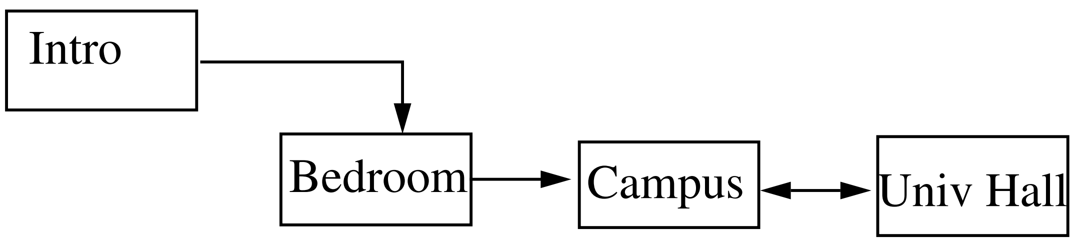
The design below of the 1993 adventure project “Quest for Vartan Gregorian” is prototypical of adventure games and more generally of Hypercard documents that involve both domain description and some Hypertalk scripting. The design starts from a high-level domain description such as that of section 4.1. To flesh out the details of the brief description of the adventure scenario, we give specifics of the characters, objects and tools of this adventure game as well as a high-level map:

Domain: the Brown Campus and Thayer St
Goal: to get an interview with Vartan Gregorian
Principal character: you, the player
Other characters: Vartan, secretary, beggar, bully
Objects and tools: wallet, pocket, key, bagel, money

The following Hypertext map has an optional introduction Its action starts in the bedroom (when you wake up in the morning) and continues on the campus and at University Hall. This design of the main locations can be refined by describing sublocations for each location area:

introduction: several descriptive Hypercards
bedroom area: bedroom, bathroom, shower
Campus: Ratty, Thayer St, post office, bookstore
University Hall: Vartan's office, secretary's office, room with safe

.



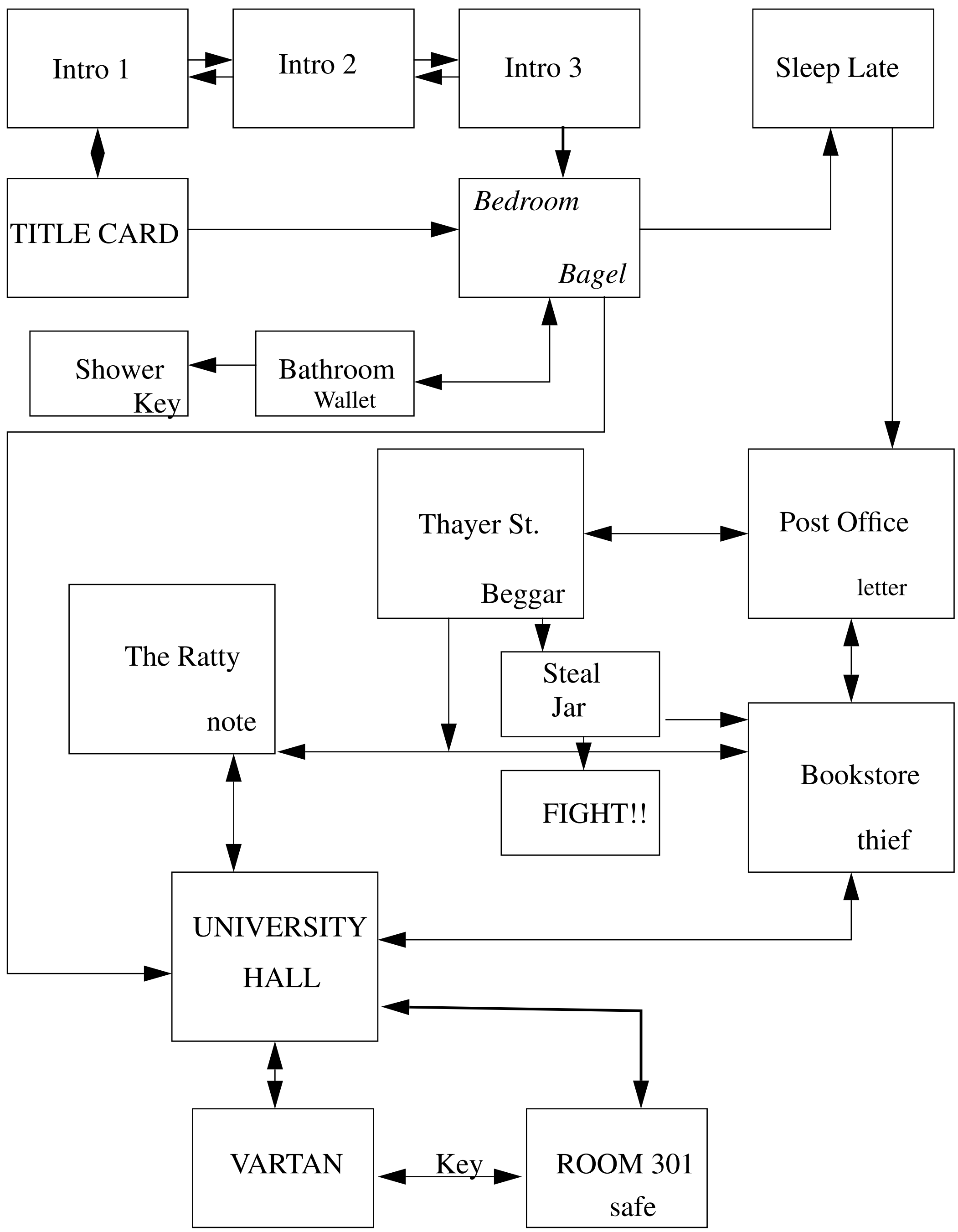
The structure of the Hypertext may be described with a more detailed map on the next page. When you wake up you can shower, sleep late, or go directly to the campus. It turns out that going to the shower will prove valuable, but there is someone in the shower and patience by waiting in the bathroom is rewarded, Do not pick up the wallet in the bathroom, it belongs to someone nasty. But pick up the key in the shower, it will come in useful later.

The post office has a message that will be useful later. The Ratty has a note on the floor, picking it up will be useful later. Thayer St has a beggar, be nice to him. The Bookstore has a thief, be careful and report him. There is a second beggar near the bookstore, be nice to him too.

At University Hall you must schedule an appointment and contribute money to the Capital Campaign before you can see the President. If you do not have enough money, you must try to find some or earn some. It turns out that room 301 has a lot of money in the safe (the Capital Campaign contributions). You need a key and the safe combination to get at the money, but if you have these then getting money from the safe to contribute to the Capital Campaign is okay, at least in this game.

This verbal description (storyboard) of the highlights of this adventure game is incomplete, but more than enough to give both your TA and yourself a feeling about how the implementation will look. You will need to do a little scripting to make this game interesting using features of Hypertalk like random numbers and animation. The TAs will help you with reasonable scripting, since the goal of this assignment is to design and implement a substantial document rather than to learn more programming.

Map of Adventure Game "Quest for Vartan Gregorian"



4.4 Student Projects in 1994

About 30 of the 85 students in the course selected the spreadsheet project as their final project, largely because it was the most practical. About 20 selected an education/information project, about 10 each selected adventure, Hypertalk, and independent projects, and four selected Mosaic.

The spreadsheet project required the creation of a collection of linked spreadsheets for a coffee shop that were typical of those required in any small busines.: The design phase consisted of specifying formats of the eight required spreadsheets and of the links between them. Though this was less substantial than the design for education, adventure, and independent projects, it was nevertheless useful in forcing students to think about the structure of their project before implementing it.

Conceptually, the collection of records of a business has a visible interface to the customer represented by the cash register and a much larger hidden infrastructure that includes information about suppliers, inventory, employees, real estate, etc. The set of spreadsheets and charts of the spreadsheet project are a simplistic model of a typical coffee shop that provides a feel for models of real businesses and familiarity with the tools used in computer-based record keeping.

.

Design of the education/information and adventure projects required a storyboard and a map like that of the adventure game of section 4.3. The topics were quite varied and included the following:

A friendly guide to American geography
  helps 8-12-year-old children learn geography, point to San Francisco to see Golden-Gate Bridge
The brain
  how the brain works and what it looks like, great visual images of the cerebellum etc
Neville Chamberlain and British appeasement
  play the role of Chamberlain in the 1930s, historical research with period pictures
How to give a full body massage
  with pictures and animation, click on body parts for animated explanation
Psychology of conformity
  experiment that tests your views on this subject, interactive prompting and analysis
Career paths in chemistry
  how different topics in chemistry relate to careers, payoff of chemistry education
Understanding eating disorders
  learn about the disorders of anorexia nervosa, bulimia, compulsive eating, health care literature
Tourism in Ghana
  education about Ghana that is helpful in planning a tour, local color
Insiders' guide to New York
  everything you should know about New York, including off the beaten track, off color
Day in Prague
  multiple-language guide to Prague, views of the castle
Play with clay
  how to create things out of clay, great images of ancient and modern artifacts
Magical lingo
  why magic words and speech are so popular

Adventure projects are typically more interactive than education/information projects, being concerned with achieving a specific goal by the player/user.Some adventure games provide a framework for learning about the domain in which the adventure takes place and can be effective as education projects. The 1994 adventure projects included the following:

A Startrek adventure
  you must save the ship when an amoeba endangers the Enterprise and everyone is aging rapidly
The search for peace and love
  women must conquer their violent male counterparts to create a better world

Chivalry
you are a knight who must find a cure for an epidemic plaguing the land
A taste of Chicago
you must perform a number of tasks in Chicago and get home safely
Graduation with honor
you must redeem reckless spring weekend actions to graduate
Helping your best friend
your friend has been framed for cheating on a test, you must prove his innocence

The design requirements for independent projects were stricter than those for the other categories because less guidance was provided by the project descriptions and correspondingly more specification was therefore required by the student. The independent projects were on the following topics:

Archeology of Mesopotamia
review of sites, history, finds, pictures, sound interview with the course instructor
Brown on-line dating service
match students based on attributes, script for creating new cards from questionnaires
Music theory for beginners
teach people to read music, click on notes to hear their sound, compose your own tune
Designing robots for the game “battletech”
includes scanned pictures of robots
Issues and algorithms of computer security
includes simulation of a scrambling algorithm for decoding
Basketball simulation
includes the roar of the crowd and pictures of Celts and Lakers
Computer users buyers’ guide
interactive with scanned pictures
Exxon Valdez oil spill
education/adventure for a real environment disaster

4.5 Mosaic Final Projects

Mosaic is more powerful than Hypercard in supporting links to remote documents specified in the markup language HTML, which determines a particular document type of the SGML markup language. Markup languages provide interfaces for documents that allows the documents and their components to be flexibly accessed from a variety of external sourcess.The following high-level project description on US presidents, which we supplied as an example to students, is on a subject on which the network is likely to contain many existing documents, and is therefore a suitable topic for a Mosaic document.

Example mosaic scenario: US Presidents
Brief description: This Mosaic project examines US presidents in the 20th century from Teddy Roosevelt to Bill Clinton. It focuses on their most important presidential decisions, how they affected the course of history, and how they are similar to or different from one another. It locates relevant archives accessible via Mosaic, ftp, gopher and create a hypertext that organizes information about 20th-century US presidents in an innovative way.

Mosaic provides access to other archives through ftp, gopher, newswatcher software, allowing the use of a wide variety of already existing information on the net and immediate access to your work by others. However, Hypercard provides more flexible support for local data and greater control of your own multimedia effects. Hypercard is more interactive and programmable, and can be used on Macs not connected to a network by simply loading ones own floppy. These trade-offs may change as systems that combine network surfing with Hypercard programming functionality are developed, but right now there is no one systems that combines the interactive control of Hypercard with the network versatility of Mosaic.

Mosaic documents can include text, visual, and audio components, and we had several help sessions to show students how to include such multimedia features in both Mosaic and Hypercard documents. There are interesting parallels between text editing and audio or visual editing that allow those familiar with text editing to catch on to multimedia editing relatively quickly.

Because Mosaic projects required learning an entirely new document preparation system there were only four Mosaic projects during 1994. But these projects were among the most interesting, and reflected the greater amount of time spent by students on the projects:

Designing a school
  restructuring a school design project from Ted Sizer's course, connecting to education databases
The Christian Bible
  the Bible from a Christian point of view, Leonardo's Last Supper
Artist of the month:
  top songs of each month with links to the song
U2
  in-depth history and albums of the band, almost complete set of recordings

Mosaic projects have an extra educational dimension because students must think about how their documents are used. This raises some of the social and legal questions considered in the course over and above the technical issues of producing the document.

Multimedia expertise for scanning and images is useful in appreciating and working with visual images in other courses. This was dramatically illustrated in the 1994 course by the student who had enhanced an image of Leonardo Da Vinci's Last Supper for his Bible project and then immediately used this technique to enhance pictures taken by an electron microscope for a biology class.

4.6 The Final Exam

The final exam provided an incentive for the review of material covered after the midterm exam. It covered the following topics:

Hypertalk:
  coffee-shop assignment, animation, random numbers
Artificial intelligence
  can machines think?, Eliza
Economics, ethics, security, wiretapping
  material covered in the lectures
Social impact:
  lectures, assignment, last four chapters of textbook
Document engineering, software engineering
  specification, design, implementation, enhancement, reuse, document life cycle

The first three questions covered loops, animation, and information transfer in Hypertalk. The fourth question on Eliza required a reading knowledge of Lisp:

Question 4. Describe briefly how Eliza fools people into thinking it is intelligent. Illustrate your answer by describing how the following Eliza dialog is executed by the Lisp program below (C is the computer and P is the person in the dialog).

C: SPEAK UP
P: I AM WORRIED ABOUT THE FINAL EXAM
C: HOW LONG HAVE YOU BEEN WORRIED ABOUT THE FINAL EXAM
P: SINCE THE COURSE ON COMPUTERS BECAME TOO HARD
C: DO MACHINES FRIGHTEN YOU
P: NO
C: PLEASE DO NOT BE SO SHORT WITH ME
P: ARE YOU BEING RUDE BECAUSE I AM NOT FRIGHTENED OF YOU
C: I AM SORRY OUR TIME IS UP GOODBYE

LISP Program that executes the ELIZA script:

```
(DEFINE (DOCTOR)
 (PROG(L S)
 (PRINT ‘(SPEAK UP!))

 LOOP
   (SETQ(S (READ))

   (COND ((MATCH ‘(I AM WORRIED *L) S)  (PRINT
    (APPEND ‘(HOW LONG HAVE YOU BEEN WORRIED) L)))

    ((MATCH ‘(* COMPUTERS *) S)
    (PRINT ‘(DO MACHINES FRIGHTEN YOU)))

   ((OR (MATCH ‘(NO) S) (MATCH ‘(YES) S))
    (PRINT ‘(PLEASE DO NOT BE SO SHORT WITH ME)))

     (T (PRINT ‘(I AM SORRY OUR TIME IS UP))
    (RETURN ‘GOODBYE)))

  (GO LOOP)))
```

The remaining four questions concerned economics, intellectual property, design, and augmenting human intellectual abilities:

Question 5. Explain the reasons for the decline of IBM and the rise of Microsoft in the 1980s. Was this decline an inevitable result of changing technology, or was it due to strategic errors of IBM that could have been avoided?

Question 6. What is intellectual property and how does it differ from physical property? Is it reasonable for Microsoft to claim Excel as its intellectual property? Briefly indicate the main arguments for and against the position that intellectual property can be owned and commercially traded. What is your opinion on this matter?

Question 7. Explain the difference between design and an implementation. Illustrate your answer by indicating the difference between the design for your final project and its implementation.

Question 8. Computers augment our intellectual abilities, just as other machines have augmented our physical abilities. Identify three ways in which computer techniques such as those discussed in this course can augment your intellectual abilities. Indicate for each why and how computers will increase your abilities.

Since the primary goal of this exam was to provide an incentive for review of course materials, its scope was indicated in detail in a lecture/review session and an evening review session run by TAs. The level of performance on the exam was high, indicating that students had actively reviewed the course materials, thereby consolidating their knowledge of Hypertalk and the material on economics and social impact.

The question on augmenting our intellect was given as much to find out what students thought important as to test their knowledge. Students commented favorably on the fact the concluding question effectively enabled them to express their opinions on the importance and usefulness of the topics covered in the course. Some of the answers of students are given below:

1) Using spreadsheets allows us to store and manipulate tabular data. It also allows us to represent data in many different ways by using graphs and charts. With this useful tool, we can model data in a much more efficient and useful way. In our Excel final project, we learned how spreadsheets can be used to model small businesses like those owned by some of our parents.

2) Email has revolutionized the way we communicate. Over the internet, we can send email to people all over the world at a moments notice. This is a great boon to communication and allows us to send all kinds of files like pictures, data and of course personal correspondence to people who are on the network.

3) Hypertexts provide a powerful new tool to manipulate texts. Instead of the linear approach, we can connect ideas or related topics by Hypercard links so that a user may browse through the information in a nonlinear manner, thus allowing far more flexibility and opportunity to explore subjects than a linear text has allowed us to in the past.

4) Word processing helps in writing better papers because it allows us to concentrate on the substance of what we write without worrying about its form. Spelling checkers help to correct spelling, formatting helps in organizing the written text, and editing helps us to easily change text when we think of better ways of expressing ourselves.

5) The internet connects us to other people and to distant documents, turning the world into a global village. We can look at art in London's art museums, browse through libraries and multimedia documents, and compose our own documents like those in the final projects.

4.7 Conclusion

Recent changes in technology and modes of computer use suggest that first courses in computing for undergraduates should change their emphasis from programming and software engineering to document management and document engineering. This course aims to realize this paradigm change, blending computer literacy, conceptual substance, and hands-on experience while covering the following topics:

introduction to the Macintosh and Hypercard
hardware, software, and algorithms
word processing, spreadsheets, and databases

networks and cyberspace
artificial intelligence
Hypertalk programming
social impact
document and software engineering

From the point of view of course design, we can view these topics as parameters that can be tweaked in both content and depth of coverage. For example, we will probably increase the amount of time devoted to spreadsheets the next time the course is taught, change the quality of the coverage of Hypertalk programming, and reduce the amount of work in the assignments for "how computers execute programs" and "can machines think?" by asking students to extend a partial stack in accordance with principles of software and document engineering.

The techniques learned in courses of this kind can be immediately applied to enrich the overall undergraduate learning experience. This clearly applies to the use of word processing, but is true also of many of the other techniques such as spreadsheets, visual image enhancement, and incremental document development, which may be used to improve the content as well as the appearance of term papers and other student work. Though much of this material is currently taught in schools, we believe that a systematic introduction to document-based information technology belongs in universities and will be widely accepted as a prerequisite, corequisite, or substitute for introductory programming courses within five to ten years. As computer usage becomes more pervasive, computers will increasingly augment day-to-day human mental abilities just as machines have increasingly augmented human physical abilities over the past 200 years.

Appendix 1: Macpaint Competition Prize Winners

Prizes were awarded in four categories:
Best Artistic:
  First prize: Boeing 747 Over China, I-Min Mau
  Second prize: Wolf, Sarah Cunningham
  Third prize: Face, Ammar Fakeeh

Best Technical:
  First prize: The Endeavor, Newport 1993, Rebecca Wilkin
  Second prize: Vegetable Still Life, Jennifer Koo
  Third prize: The Skier, Keith Giordano

Funniest:
  First prize: Bobbitt, Antonio Cabral
  Second prize: Window Dressing, Elizabeth Fry
  Third prize: Mirror, Mirror on the Wall, Claire Gorman

Most Original:
  First prize: Seeing Eye Frog, Paul Shieh
  Second prize: Abstract Objects, Melanie Allen
  Third prize: Apathy/Anxiety, Kara Juneau