A Model Curriculum for K-12 Computer Science*

Level 2 Objectives and Outlines

Anita Verno
Bergen Community College
400 Paramus Road
Paramus, NJ 07652
201-447-7909
averno@bergen.edu

Debbie Carter
Lancaster County Day School
725 Hamilton Road
Lancaster, PA 17603
717-392-2916
carterd@e-lcds.org

Robb Cutler
The Harker School
500 Saratoga Avenue
San Jose, CA 95129
robbc@harker.org

Michelle Hutton
The Girls' Middle School
180 N. Rengstorff Ave.
Mountain View, CA 94043
mfh@pobox.com

Lenny Pitt
University of Illinois at Urbana-Champaign
201 North Goodwin Ave
Urbana, IL 61801
217-244-6027
pitt@uiuc.edu

The authors would like to express appreciation for the feedback from the individuals listed in the Acknowledgements section. We would specifically like to recognize the contributions provided by Dan Frost, Donald Bren School of Information and Computer Sciences, University of California, Irvine, for his assistance with the organization and wording of Appendix A of this document.

*Tucker, Allen, (editor), Deek, F., Jones, J., McCowan, D., Stephenson, C., and Verno, A. *A Model Curriculum for K-12 Computer Science: Final Report of the ACM K-12 Task Force Curriculum Committee.* Association for Computing Machinery (ACM), New York, New York, October, 2003. http://www.acm.org/education/k12/k12final1022.pdf

Table of Contents

Introduction

Overview:

The ACM K-12 Computer Science Model Curriculum was developed in 2003 to provide a broad outline from which a K-12 computer science curriculum can be developed. The Model Curriculum was a response to the view that computer science education is not clearly defined or well-established at the K-12 level. A national computer science curriculum which stabilizes the objectives and content of high school CS has implications beyond K-12 education. It will assist students with further study in computer science / information technology / information systems / engineering and will help with the larger national effort to build America's position as a global leader in technological knowledge and expertise.

This document continues efforts by the ACM and Computer Science Teachers Association[*] (CSTA) to create a comprehensive body of resources to support the implementation of a national K-12 computer science curriculum. The Model curriculum provides an overview of computer science content broken into four levels: Level 1, recommended for students in grades K-8; Level 2, recommended for students in grades 9 or 10; and Levels 3 and 4, recommended for students in upper grades. The Level 1 curriculum closely follows ISTE standards and is therefore a fairly well-developed set of learning objectives with available resource materials. The Level 2 curriculum, suggested for all students, was broadly described in the Model. This document specifies the learning objectives, assessment measures, and sample educational activities of Level 2, which can be used to encourage a uniform level of teaching across the US. Lesson plans and activities to support each stated objective are available in the CSTA Web Repository[†] .

Purpose and role of the Level 2 Objectives and Outlines

Level 2 of the ACM Model Curriculum for K-12 Computer Science introduces the formal study of computer science and its role in the modern world. Technology plays an increasing role in the modern world; computer science provides students with the skills and knowledge to understand the technology they use daily and to extrapolate this knowledge to understand and use emerging technologies. This document is intended to assist in the creation and delivery of an introductory course in computer science by providing activities and objectives recommended for early high school students. The teacher should modify the course based upon available technical resources and make it appropriate to the level of the students.

Despite common stereotypes, computer science encompasses more than just programming. It builds the basic logical problem solving skills and framework required for understanding an increasingly complex and technological world. Level 2 reflects the breadth of the discipline and

[*] The Computer Science Teachers Association is a membership organization that supports and promotes the teaching of computer science and other computing disciplines by providing opportunities for K-12 teachers and students to better understand the computing disciplines and to more successfully prepare themselves to teach and to learn.

[†] The goal of the Web Repository is to create a web-based multi-grade repository of appropriate materials for K-12 computer science teachers to support their teaching and professional development. Teachers will be encouraged to submit their own original materials, to the benefit of the larger community. The Web Repository remains under development as of February 2006.

provides students with many opportunities to discover an interest in computer science. The variety of activities and topics gives each student multiple opportunities to develop an appreciation of computer science.

The ACM Model Curriculum for K-12 Computer Science employs a spiral model in which students are introduced to topics, skills, and concepts repeatedly, at a higher level each time. Level 2 is an intermediate course between the introductory level in elementary and middle school and the more rigorous demands of Levels 3 and 4. We recommend that teachers read the entire ACM Model Curriculum for K-12 Computer Science before implementing these Objectives and Outlines in order to understand how this document fits in the larger context. Level 2 may be the first or the final computer science course taken by many students. Teachers should choose activities at a level of rigor appropriate to the prior experience of the students.

Ideally, Level 2 is a year-long standalone course taught by a qualified computer science teacher. However, we recognize that the constraints of many schools may render this recommendation impossible to meet. Some of the course content may be presented quickly in a semester-length course. The balance may then be integrated across disciplines. The course can also be totally presented in a project-based integrated format. Where integrated, teachers must employ team-teaching, with a computer science teacher presenting the CS content and a subject-area teacher covering the underlying subject-matter content.

About this Document

This document can be used to help define standards and help teachers teach to those standards.

The Level 2 Objectives and Outlines consists of 14 individual topics. Most of these topics are listed in the ACM Model Curriculum for Computer Science. Certain larger content areas were broken into smaller units, and some smaller units were combined into larger areas of study. Some topics, such as computer history, are not addressed in the document, but may be added by individual teachers as time and interest warrant. Other concepts, such as security, are addressed within several topics, providing a practical base for understanding these complex issues. Within each topic are goals, objectives, and proposed activities. The numbering scheme for topics corresponds to resources located in the CSTA Web Repository, making it easy to find a variety of activities relating to each topic.

Level 2, as with the entire ACM Model Curriculum for K-12 Computer Science, was informed by state standards and textbooks in addition to the varied feedback received. It is not intended to replace textbooks or other resources currently available, but to provide a curriculum structure and guide. We hope that it will guide the development of future textbooks and other curricular resources. At the end of this document is an articulated list of resources particularly used in creating the objectives and outlines. By the time you read this document, some of the resources listed will be outdated. However, we believe that the Level 2 document is flexible enough that the core program of study recommended will stand the test of time.

For this reason, no programming languages or platforms are specified in this document. To the extent possible, no specific software is recommended. Teachers should choose activities and materials appropriate for the students and for the technical environment in which they teach. This document provides an outline of skills and concepts which can be introduced in a variety of ways; in many cases with current hardware and software, or with minimal infusion of financial resources for new technology purchases.

How to Use this Document

This is a thinking person's curriculum. It should not be followed blindly, but used as a guide. Each teacher should craft the curriculum to the needs of his or her students, covering the material at an appropriate depth and speed.

Although each topic is presented independently, many are interdependent. Teachers are encouraged to link topics as appropriate. For example, Topic 9: Ethical Issues is broadly applicable across many areas such as Topic 4: Internet Concepts. Additionally, although the topics are numbered, the numbering does not reflect a recommended sequencing for presentation. Rather it provides a method for locating and downloading resources from the CSTA Web Repository.

Therefore, this document is flexible and can be used in multiple ways. A teacher may choose to begin with Topic 1: Principles of Computer Organization and proceed straight through the document, teaching each topic in order. A teacher may choose to present the topics in a different order based on schedule requirements, prior knowledge of the students, or a sense that certain complimentary topics should be presented together. The objectives can also be completely reorganized and presented through a project-based model. Dan Frost of the University of California at Irvine assisted the Level 2 Committee in creating a second organizational approach to the topics. Appendix A associates each focus with one or more specific facets of CS.

Approximate working times for each topic are listed in the document. However, teachers should use their discretion based on the previous knowledge of their students, the resources available, and the depth to which they wish to cover the material.

Each focus area has one sample activity listed. This activity is intended as an example of the depth of detail suggested by the Committee. Many activities at different levels of difficulty are available through the Web Repository[‡]; a collection of instructional resources submitted by computer science teachers.

[‡] The goal of the Web Repository is to create a Web-based multi-grade repository of appropriate materials for K-12 computer science teachers to support their teaching and professional development. Teachers will be encouraged to submit their own original materials, to the benefit of the larger community. The Web Repository remains under development as of February 2006.

Topic 1: Principles of Computer Organization

Topic Description:

Principles of Computer Organization will introduce the student to the major components of the computer including: input, output, memory, storage, processing, software, and the operating system.

Textbooks and Supplies:

A computer that can be opened up and taken apart; different types of computers and operating systems recommended. Printed advertising for computers, or a computer with Internet access for web research.

Time to Complete: 2 weeks

Student Learning Objectives	Assessment Measures
The student will be able to:	
1. Identify the various functional components of a computer.	Lab activity Written activity
2. Match a list of computer terms and definitions/functions.	Written activity
3. Describe the interaction of the various functional components of the computer.	Lab activity Written activity
4. Make appropriate decisions when purchasing a computer for home use.	Written activity

Assessment Recommendations: An average of 60% from combined assessment measures is required to demonstrate proficiency in course material.	
Lab activities	50%
Written activities, including tests, quizzes, and written assignments	50%

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
1. Terminology	Identify, define, and appropriately use the key terms associated with the computer and its components.
2. Identifying hardware components	Where possible, provide each student with the opportunity to take apart an old computer and locate and identify the various components.
3. Identifying software components	Discuss the role of system software and application software.
4. Describing the interaction of components	Students perform a role play where each student actor represents one component – hardware or software. Scripts describe their general actions; the teacher provides the data for the interactions.
5. Purchasing a computer	Students locate computer advertisements in print or online. A comparative table is created that lists the advantages and disadvantages of at least three advertised computers for possible personal use.

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
6. File systems and organization	Explain directory structure. Students create and use nested directories. Students should explain the advantages and disadvantages to a flat structure vs. a hierarchical structure methodology.
7. Diagnose & troubleshoot PC problems	Discuss common PC problems and solutions. Have students experience a practical lab where they must diagnose and fix a set of "broken" PCs (disconnected network cables, unplugged monitor cables, moved/deleted shortcut icons, etc.)

Topic 2: Problem Solving

Topic Description:

This topic covers the basic steps in algorithmic problem-solving, including the problem statement and exploration, examination of sample instances, design, program coding, testing, and verification. Tools for expressing design will be used.

Textbooks and Supplies:

A programming language; interactive development environment recommended.

Time to Complete: 1-2 weeks with continual reinforcement as appropriate

Student Learning Objectives	Assessment Measures
The student will be able to:	
1. Name and explain the steps in the problem-solving process.	Written activity
2. Solve a problem by applying the problem-solving process.	Written activity
3. Express a solution using standard design tools.	Written activity Lab activity
4. Determine if a given algorithm successfully solves a stated problem.	Written activity

Assessment Recommendations: An average of 60% from combined assessment measures is required to demonstrate proficiency in course material.	
Written activities, including tests, quizzes, and written assignments	80%
Lab activities	20%

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
1. Problem-solving process	Students are given simple problems to solve. They write the process they used to solve each problem, defining a general problem-solving strategy.
2. Understanding the problem (analysis)	Students will be given several problems to review. Some will have all the detail stated within the text. Others will be missing detail. For each complete problem, students will restate the problem, including input, output, formulas, and other relationships between the input and the output. For each incomplete problem, students will request an interview with the problem owner to ascertain the complete problem and reformulate it into a complete statement.
3. Exploring problems: problem representation, problem-solving heuristics and strategies.	Students employ the use of various forms of representation (e.g., patterns, grids, graphs, sets) to understand problems. Solutions are explored that can make the problem simpler by considering special cases, smaller cases, solving sub-problems, or searching for patterns.

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
4. Problem design	Students employ a design methodology (e.g., top-down, bottom-up, combination) to a restated problem and create an algorithm. The algorithm is expressed using a standard tool such as flowcharts, pseudocode, Unified Modeling Language (UML), and Nassi-Shneiderman diagrams.
5. Problem data	Students evaluate a problem and determine an appropriate set of data that can be used to produce correct results.
6. Solution accuracy	Students perform a structured walk-through of their algorithm to ensure it solves the problem and make corrections as needed. Students trade solutions and perform a structured walk-through of each other's algorithm.
7. Program coding and testing	Where possible, students code the problem to determine if it works as anticipated.
8. Re-evaluation and refinement	Students identify errors and propose fixes via re-application of the problem solving process.
9. Communicate results	Students document their solution and share with other classmates.

Topic 3: Basic Components of Computer Networks

Topic Description:

Basic knowledge of networking will be introduced as a building block for understanding how computers communicate. The student will become familiar with the basic components of a computer network including servers, file protection, queues, routing protocols for connection/communication, spoolers, shared resources, and fault-tolerance. Computer security and protocols will be discussed.

Textbooks and Supplies:

As needed to demonstrate networking concepts.

Time to Complete: 2 weeks

Student Learning Objectives	**Assessment Measures**
The student will be able to:	
1. Match a list of networking terms with their definitions.	Written activity
2. State the hardware requirements for adding a computer to a network.	Written activity
3. Connect a computer to a network.	Lab activity
4. State at least three security issues that can affect a computer that is connected to a network.	Written activity
5. Describe at least two network protocols and state the reason to select one over another.	Written activity

Assessment Recommendations: An average of 60% from combined assessment measures is required to demonstrate proficiency in course material.	
Lab activities	70%
Written activities, including tests, quizzes, and written assignments	30%

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
1. Terminology	Students define a network and various networking terms.
2. Data transfer	Each student acts as a computer on a network using a designated network configuration. One student sends an object (data) across the network to another. Discuss how the second student knows the data belongs to him/her. Have the first student pass 20 heavy books across the network. Can they all be passed at once? Explain the concept of packets.

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
3. Data collision and network failure	Continuing the enactment from above, have a second student attempt to pass data at the same time as the original data is passing his/her connection, causing a collision. How can this be avoided? One student hurt his/her arm and can no longer pass the object. Discuss the possible problems and protections in modern networks. Students try several configurations, noting the effect on collision, fault tolerance, number of supported connections, and number of links from origin to source (points of insecurity).
4. Connecting a computer to a network	Discuss the hardware and software needed to connect a computer to a network.
5. Identity	Demonstrate, if possible, that every computer on a network has a unique address. Compare the address to a home phone number or house address. Discuss hierarchical addressing using examples like countryCode.areaCode.exchange.suffix, as compared to IP addresses and dynamic IP addresses compared to local extension within phone network.
6. Applications of networks	Discuss and list networks with which students have experience (e.g., phone, Internet, school LAN). Discuss the number of connections and the different purposes of each. Include specific applications of a network such as video conferencing and phone conferencing.
7. Mobile computing	Discuss wireless computing and specific applications that use wireless computing (e.g., PDA, handheld, laptop). Introduce mobile computing concepts and, as available, demonstrate the use of mobile devices.
8. Network Security	Lead students to a discussion of protecting privacy and security of passwords, etc. Who is responsible for maintaining important data? Discuss ways to protect a networked computer from outside threats.

Topic 4: Internet Concepts

Topic Description:

Internet concepts will allow students to consider how Internet elements (e.g. email, chat, WWW) are organized, will engage students in effective searching, and will focus on productive use of e-mail.

Textbooks and Supplies:

Internet access; software to demonstrate Internet elements

Time to Complete: 1-2 weeks

Student Learning Objectives	Assessment Measures
The student will be able to:	
1. List at least three strengths and weaknesses of each of three Internet elements and at least one use for each.	Lab activity Written activity
2. Use at least two Internet elements.	Lab activity
3. Use appropriate tools and methods to execute Internet searches which yield requested data.	Lab activity
4. Develop and use a rubric to evaluate the results of web searches and reliability of information found on the web.	Lab activity

Assessment Recommendations: An average of 60% from combined assessment measures is required to demonstrate proficiency in course material.	
Lab activities	70%
Written activities, including tests, quizzes, and written assignments	30%

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
1. Internet elements	Students use a variety of Internet elements, including those used primarily for one-way communication, private and group discussion, and collaboration. Students articulate the benefits and limits of different Internet elements and identify appropriate uses for each. Ethical and legal concerns should be discussed as desired.
2. Search engine fundamentals	Explanation of what a search engine is and how it functions. Cover similarities and differences between search engines such as advanced search commands, organization of output, whether they are influenced by hidden factors such as money. Compare identical searches in various search engines.

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
3. Search engines and directories	Brainstorm Web-based resources and categorize them. Compare and contrast their features. Try to find identical/similar information using a search engine and a directory. Articulate strengths and weaknesses of each and when it would be appropriate to use each.
4. Refining search parameters	Perform searches and use various methods to refine the search. Include advanced search features; Boolean operators; factors that affect results, such as spelling, wildcards, and quotes; use of different strategies, including keyword search. Compare results to expected results and continue to refine until expected results are met. Express desired document characteristics using logical (AND/OR/NOT) or set (union, intersection) operations.
5. Evaluating Web sites	Create an evaluation tool to use in determining reliability and quality of individual Web sites. Consider authorship, whether or not the author is an authority and/or has credibility. Evaluate several Web pages from authorities to bogus resources such as the committee to ban dihydrogen monoxide.
6. Security on the Internet	Encryption: The teacher creates several different encryption keys. Students encrypt messages to one another using a designated key. The message is sent to another student to be decrypted, but the key is not. Discussion follows regarding protection when providing personal identification information like social security numbers and credit card numbers.

Topic 5: Hierarchy and Abstraction in Computing

Topic Description:

The notions of hierarchy and abstraction are central to computing. They are crucial to the translation between machine code and a user-friendly interface, to creating reusable code, and to the design of software that is broadly applicable rather than solving only a narrowly defined problem. This unit makes these abstract ideas concrete by focusing first on real-life (non-computing) examples, and then on the specific uses of hierarchy and abstraction in computer science.

Textbooks and Supplies:

A programming language; interactive development environment recommended.

Time to Complete: 2 weeks

Student Learning Objectives	Assessment Measures
The student will be able to:	
1. Articulate three levels of hierarchy within an object or activity	Written Activity
2. Given a diagram showing how source code becomes an executable application, state the activity occurring at each step and the intermediate files produced.	Written Activity
3. Articulate the difference in time and amount of coding when using a higher-level language vs. assembly/machine language.	Lab Activity Written Activity

Assessment Recommendations: An average of 60% from combined assessment measures is required to demonstrate proficiency in course material.	
Written activities, including tests, quizzes, and written assignments	70%
Lab activities	30%

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
1. Decomposing the complex	Students are asked to describe the parts of a car. Discussion results in the description of at least three layers of a car object such as JohnCar.door.front and JohnCar.door.rear; JohnCar.seat.front and JohnCar.seat.rear. Students then state the similarities of properties and actions at each level and determine why the next level is necessary. Appropriate notation should be used. This discussion should also include file structure hierarchy and various notations for levels i.e. slash vs. dot. Using the same technique, students then describe the components of a baseball game providing at least two layers such as MyTeam.outfield.left and MyTeam.outfield.right and MyTeam.infield.first and MyTeam.infield.shortstop.

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
2. Translating from source code to execution	Discuss how higher-level languages allow humans to code using familiar concepts rather than by dealing with machine language. Provide a diagram of the process of translation from high level code to low level assembler or machine code. Introduce terminology such as interpreter and compiler during a discussion of the steps in the translation. Optionally, students role play the process of translation.
3. The role of circuits	Students simulate the execution of a simple machine instruction at the gate level: addition of two binary numbers. After introduction to gates, a schematic of a one or two-bit adder is laid out with masking tape. Students are bits with 0/1 signs, and move through the circuit, obeying the rules of each “gatekeeper.”
4. The power of hierarchy	Students apply top-down design techniques to write an algorithm for preparing a research paper that begins with three main parts: introduction, body, and conclusion. Each part is further refined into sub-steps. Point out the hierarchy by drawing a tree or similar graph. Can the algorithm be applied to a paper for a history class? How about in an English class? The ability to re-use the algorithm in a different context rather than begin again demonstrates the power of hierarchy.
5. Abstraction	Use models as an example of abstraction. Students identify a model and list what details are abstracted vs. shown. Students articulate why the abstracted details were not important. Students create a software model (e.g., a *PowerPoint* model of the digestive system, a *Squeak* model of a car, a *Flash* model of anything). They explain how the model is an abstraction of the original. Students give examples of multiple layers of abstraction in computers (e.g., if hardware is lowest level, what is next, what is next, etc., up to highest level of abstraction.)

Topic 6: Connections Between Mathematics and Computer Science

Topic Description:

Computer science is fundamentally connected to several elements of mathematics. This unit focuses on three foundational themes that are central to computation: binary numbers as related to digital computation and data representation, Boolean logic relating to conditional statements, and representations for problem modeling such as sets and functions.

Textbooks and Supplies:

A programming language; interactive development environment recommended.

Time to Complete: 2 weeks

Student Learning Objectives	Assessment Measures
The student will be able to:	
1. Convert between decimal, binary, and hexadecimal numbers.	Written activity
2. Create and translate an encoded message using a simple method of encryption.	Written activity
3. Write conditional statements that include simple and complex Boolean expressions to solve stated problems.	Written activity Lab activity
4. Convert a problem description into correct set notation and apply appropriate set operations.	Written activity
5. Write a function (method, etc.) that returns the correct value, given a function definition in mathematical notation.	Written activity Lab activity

Assessment Recommendations: An average of 60% from combined assessment measures is required to demonstrate proficiency in course material.	
Written activities, including tests, quizzes, and written assignments	70%
Lab activities	30%

Detailed Outline

Focus	Sample Lab / Hands-on Activity
1. Binary number system	Presentation about how data and instructions are stored in binary code, including conversions between decimal, binary and hex. Students work additional problems at their seats and/or at the board, perhaps as part of a game.
2. Understanding various forms of encryption	Using a simple form of encryption, encrypt a text message and pass it to a classmate, along with a key. The receiver will use the key to translate the encrypted message back to the original text. Discuss the reasons why data is encrypted.

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
3. Conditionals and Boolean logic	Students implement a movie rating system that assigns a rating (from 1 to 100) based on the user's answers to a series of questions (e.g., "Does it involve a car chase?" "Is there romance?" "Are there teenagers?"). If available, the movie rating system is coded. Students use cards with whimsical characters defined by several Boolean attributes (hairy/bald one/two eyes, etc.) One team chooses a hidden logical rule such as "(hairy AND one-eye) OR tall". The other team chooses a card and guesses whether it satisfies the rule, with the goal of getting the fewest incorrect guesses until the rule is inferred and no further mistakes occur.
4. Functions including parameters and mathematical notation	Students practice writing functions and methods.
5. Set representation, and set operations	Students use whimsical binary character cards to populate a three-ring Venn diagram with circles corresponding to three attributes (E.g., hairy, two-eyes, short). Challenge questions are to describe regions and combinations of regions using logical notation (e.g., "hairy OR short" for the union, "(hairy AND short) OR (hairy AND two-eyes) OR (short AND two-eyes)". Similarity of union and OR, intersection and AND, complement and NOT should be stressed.

Topic 7: Models of Intelligent Behavior

Topic Description:

The primary goal is to demonstrate that “intelligent” machine behavior is not “magic” but is based on algorithms applied to useful representations of information. A secondary goal is for students to appreciate those characteristics that make certain tasks easy or difficult for computers, and how these differ from those that humans characteristically find easy or difficult. Intelligent software is becoming increasingly prevalent.

Textbooks and Supplies:

Board games, Internet access, etc. as needed depending upon labs, demonstrations & activities selected.

Time to Complete: 1-2 weeks

Student Learning Objectives	**Assessment Measures**
The student will be able to:	
1. Given a list of tasks from several application areas of artificial intelligence, indicate whether or not computers can do those tasks, using current technology.	Written Activity

Assessment Recommendations: An average of 60% from combined assessment measures is required to demonstrate proficiency in course material.	
Written activities, including tests, quizzes, and written assignments	80%
Lab participation	20%

Detailed Outline *(Select activities based on student needs and available resources.)*	
Focus	**Sample Lab / Hands-on Activity**
1. What is Intelligence?	Teacher provides Web sites and material that will help students research and debate this question.
2. Natural Language	Syntax: Students play a game of “Mad Libs." Discussion: Are the resulting paragraphs syntactically correct? Can meaning be extracted? Meaning: Students examine headlines or other short passages in which the literal meanings differ from the meanings that would be understood by humans (e.g., “Kicking Baby Considered to be Healthy” and “Man Shoots Neighbor with Machete.”) and then create some of their own. Discussion: Can a computer be taught to infer something that is not stated in the text, ignoring a literal interpretation if necessary?

Detailed Outline
(Select activities based on student needs and available resources.)

Focus	**Sample Lab / Hands-on Activity**
3. Knowledge-based Systems	Students explore some ways that knowledge can be represented and put to use, including rule-based deduction and tree-based classification. Students engage in a game of deduction using existing (teacher-specified) rules to deduce new facts. For example, using rules based on a student's grade level and native language, assign her to a foreign language course. Present everyday examples of expert systems such as medical diagnostic screening and car repair. Students draw decision trees to represent portions of a system, illustrating the how responses to specific questions lead to other questions and eventual results.
4. Machine Learning	Students discuss simple learning algorithms employed by online retailers where previous purchases lead to suggested purchases. Each student creates a simple learning algorithm to predict a classmate's likes/dislikes of songs or clothes based on relevant features. Discussion: programs can be written to observe patterns of behavior, resulting in modification of a program-controlled activity and its knowledge base.
5. Game Playing, Problem Solving, and Searching	Students play game of Nim (or other game with simple guaranteed strategy) with each other and against the computer. The winning strategy is discussed. Students play games against a computer opponent (Othello, chess). How can a computer make decisions about each move? Problem Solving and Tree Search: Students try to solve the 4-queens problem (place four queens on a 4 x 4 chessboard so that none are attacking each other). Afterwards, the teacher models search and backtrack by generating a search tree of board configurations that lead to a solution. Students employ the tree-search method to solve a logical problem like the river-crossing (farmer/fox/chicken/corn). Discussion: how "intelligent" behavior can arise from a good understanding of a problem or an exhaustive or principled search.
6. Robotics	A large selection of robotics materials at all levels exists on the World Wide Web. Select appropriate materials for your students.
7. Vision and Speech	Presentation/demo, where possible, of a speech recognition and computer vision system.
8. The Myth of Intelligent Behavior	Explore how intelligent behavior suddenly loses its luster after one understands the little man behind the curtain. Points to explore: sometimes cheap tricks go a long way (Nim). Sometimes, a lot of work goes only a little way (natural language). Sometimes, a lot of work goes a long way (chess).

Topic 8: Interdisciplinary Utility of Computers and Problem Solving in the Modern World

Topic Description:

Students will gain an appreciation for the many ways (types of use) in which computers have had an impact across the range of human activity, as well as for the many different fields in which they are used. Examples should illustrate the broad, interdisciplinary utility of computers and algorithmic problem solving in the modern world.

Textbooks and Supplies:

Computer with Internet connection, magazines, newspapers.

Time to Complete: 1-2 weeks

Student Learning Objectives	Assessment Measures
The student will be able to:	
1. Find (in newspapers, magazines, through interviews, or on the Internet) and describe three examples of the use of technology in non-computer fields.	Presentation Written activity
2. Choose the appropriate category for each item in a list of technology applications.	Written activity

Assessment Recommendations: An average of 60% from combined assessment measures is required to demonstrate proficiency in course material.	
Lab activities	75%
Written activities, including tests, quizzes, and written assignments	25%

Detailed Outline
(Select activities based on student needs and available resources.)

Focus	Sample Lab / Hands-on Activity
1. How are computers used in our world?	Student list various ways they have seen computers used. The teacher assists in categorizing them based on common use (e.g., personal finances, business, entertainment, government, education, etc.) Teacher adds examples to be sure list contains many common applications. Several uses of computers are selected for student analysis. The following types of questions are asked about each one: What is the general computational task? What is the input? What is the output? Does the task rely on an underlying body of data? What security or ethical considerations are involved? What makes this use different from the others under analysis?
2. Information storage and retrieval, including examples from the web, such as tables and searchable databases	Students participate in a scavenger hunt by visiting a variety of designated Web sites to retrieve specified information. (Ideally, this activity would be integrated with another course, so that students were using the tools to solve a problem in another discipline.)

Detailed Outline *(Select activities based on student needs and available resources.)*	
Focus	**Sample Lab / Hands-on Activity**
3. Decision-making support	Discuss how computers offer support for decision-making (e.g., knowledge bases, sales data analysis, expert systems.)
4. Data visualization	Presentation by teacher or guest speaker about uses of data visualization (e.g., CAT scan, weather prediction). If available, take a field trip to view data visualization and learn how this application assists the population.
5. Communications	Demonstration and hands-on (where possible) experimenting with various types of communications (e.g., Web page, web forum, e-mail, mobile phone, PDA, telemeeting, multimedia presentation).
6. Modeling and design, including CAD and simulations	Stage a simulation of a simple process or event, such as the transmission of a (biological) virus. This can be accomplished using computer software, PDAs, or by passing colored paper in sealed envelopes among the students (with one color being the "virus"). Why do computers excel at simulation? Select some real-world examples for discussion. Use modeling software to model a simple structure from another subject area (e.g., the structure of a molecule or a set design for a theater class play).
7. Art, music, and video for entertainment, including editing of media, graphics, games, and 3D animation	Develop a simple graphical game. Record and manipulate music or sounds. Visit Web sites that display animated graphics. Locate on the web and then list applications that permit creation and editing of images, audio, and video. Discuss advantages and disadvantages of the various products listed.
8. Education and training	Students use tutorial software to learn something new.
9. E-Commerce	Discussion of activities that fall within E-commerce such as advertising, online ordering, and auctions. Create a list of items to purchase online. Students locate the best price(s), add the item(s) to their shopping cart, and save the order.
10. Embedded systems	Discussion of example uses of computer technology embedded in various devices (auto diagnostics, pacemakers, security cameras that respond to and record motion, sensors and control, appliances). Divide students into teams. Each team lists as many examples as possible of computers (chips) working in school, businesses, and homes, describing their functions, if known.
11. More examples of computing applications	Students interview adults in their home/neighborhood and ask how they use computes at work and how their use of computers has changed over the past 10-20 years. Students should also ask how computers have changed their job during this time period. Students find three examples of the use of technology and prepare an in-class and/or written presentation, according to the classifications listed above. Resources: newspapers, magazines, Internet news sites, interviews with adults.

Topic 9: Ethical Issues

Topic Description:

The proliferation of computers and networks raises a number of ethical issues. Technology has had both positive and negative impacts on human culture. Students will be able to identify ethical behavior and articulate both sides of ethical topics.

Textbooks and Supplies:

Word processing and presentation software recommended.

Time to Complete: 1 week with continual reinforcement as appropriate

Student Learning Objectives	Assessment Measures
The student will be able to:	
1. Distinguish between ethical and legal issues in a case study by listing the issues that can be resolved through the legal system and those issues that cannot be legally resolved.	Written activity
2. Defend an ethical stance given a controversial or ethically ambiguous situation in a debate.	Written or Oral activity
3. List and explain at least two positive and negative effects of one technological innovation on human culture.	Written activity
4. Define intellectual property and state the impact of provisions to protect it.	Written activity
5. Identify at least two benefits and two drawbacks of using commercial, public domain, open source, and shareware.	Written activity
6. Demonstrate behavior in the use of technology that conforms to school and local code	School activities

Assessment Recommendations: An average of 60% from combined assessment measures is required to demonstrate proficiency in course material.	
Written activities, including tests, quizzes, and written assignments	90%
Oral activity	10%

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
1. Terminology	The difference between legal and ethical, where they overlap and where they don't. Differences between values and ethics. Foundation in philosophical ethics. Create a list of criteria for determination of whether a decision is ethical. Create guidelines for ethical use and creation of technology.

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
2. How technology has changed ethical and legal issues	Discuss the ways that technology has increased access to personal and classified information, facilitated copying, and blurred jurisdictional boundaries. Include security issues. Examine case studies.
3. The effect of technology on human culture, including historical considerations	Choose one technological innovation from a provided list and write a paper explaining at least two positive and negative effects on users and others impacted by the innovation. Writing assignment to predict how technology may affect human culture in the future.
4. Privacy and sharing of personal information	Have students use search engines to search for Web pages about themselves. Alternatively, do a pre-search for students and collect information to present. Lead a discussion about the security and privacy of information on the Internet including e-mail. Consider the implication of copyright – who owns the images and information posted on these sites? Ask students if they have secrets that they do not want to share. How would they feel if someone were able to obtain that secret by snooping in their locked secure journal? How would they feel if the person then shares this private knowledge with the rest of the class? What can they do to prevent someone from obtaining their personal information, e.g., hide the key? Discuss possible hiding places and why some are better than others. Lead students to a discussion of protecting privacy on the Internet and security of passwords, etc.
5. Intellectual property, copyright, and fair use	Create a guide for other students outlining copyright, fair use, and adherence to the school AUP within the school context.
6. Responsible use of software	Compare and contrast the responsibilities of using commercial, public domain, shareware, and open source software. What are the benefits and limitations of each?

Topic 10: Careers in Computing

Topic Description:

Careers in Computing will introduce the student to a range of jobs and careers available in the field of information technology and computing.

Textbooks and Supplies:

Internet access; Employment section of newspaper.

Time to Complete: 1 week

Student Learning Objectives	Assessment Measures
The student will be able to:	
1. Create a chart of at least five careers available in the field of information technology and computing, including the career name, educational level necessary, experience requirements, job description, and salary range.	Written activity
2. Use an online job search site to determine if there is an opening for a specified job title.	Lab activity

Assessment Recommendations: An average of 60% from combined assessment measures is required to demonstrate proficiency in course material.	
Lab activities	70%
Written activities, including tests, quizzes, and written assignments	30%

Detailed Outline

Focus	Sample Lab / Hands-on Activity
1. Job/Career Titles and Descriptions	Students use the Internet to research jobs available on job listing sites. Titles, descriptions, requirements, and, salary ranges are to be charted.
2. Guest Lecturers	Guest lecturers, from various technology-related careers, can be invited to talk with students about their jobs. Students have the opportunity to ask questions and solicit information.
3. Current Events	Students bring in articles and lead discussions on current events in the technology-related career sector.
4. Job Availability	Students use Web sites and the library to research labor statistics (local, national, and international) for a group of assigned computer careers.
5. The Perfect Job	Students write up a description of their ideal technology-related job/career.

Topic 11: Programming Languages

Topic Description:

Programming Languages will introduce the student to some basic issues associated with program design and development. The focus of this unit is to establish an appreciation of the work being done by software.

Textbooks and Supplies:

A programming language; interactive development environment recommended.

Time to Complete: 2-4 weeks

Student Learning Objectives	Assessment Measures
The student will be able to:	
1. Code, test, and execute a program that corresponds to a set of specifications.	Lab activity
2. Convert a word problem into code using top-down design.	Written activity Lab activity
3. Select appropriate data types.	Written activity Lab activity
4. Write structured program code.	Lab activity
5. Draw a series of diagrams showing the scope and values of variables during execution of a simple program.	Written activity

Assessment Recommendations: An average of 60% from combined assessment measures is required to demonstrate proficiency in course material.	
Lab activities	50%
Written activities, including tests, quizzes, and written assignments	50%

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
1. Terminology	Identify and define key terms associated with programming.
2. Representation of text inside the computer	Each student writes a sentence in binary and exchanges it with a neighbor. The neighbor translates the sentence into text. Students stand or sit to mime a secret word in binary. Flashlights can also be used to represent binary code.
3. Representation of numbers inside the computer, including the largest and smallest values which can be represented in each of several types	Numbers are placed into imaginary bytes in a grid, each imaginary byte having a unique address. (A spreadsheet can be used for this purpose.) Instructions are provided to add and subtract values by address. Some of the resulting numbers should be too large to store in the imaginary byte and will overflow.

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
4. Data types: integer, floating point, character/string, and Boolean types and appropriate operations	Evaluate the results of mathematical and logical expressions using integer, floating point, and mixed arithmetic. Evaluate the results of expressions using relational and Boolean operators. Problems should require students to understand order of operations.
5. Program execution	Draw a flowchart showing the process from source to executable, including the return flow for syntax and semantic errors.
6. Programming design techniques	Students are given a word problem and state the input required, output to be produced, and formulas required. The program flow is diagramed before the coding process begins.
7. Programming style	Distribute the written code for a program that has no comments, one-letter non-descriptive variable names, multiple statements on a line, etc. What does this program do? Have students enter and run the program to determine what it does and make necessary changes to employ good style.
8. Programming statements for output	Write a program that displays output.
9. Declaring and using constants and variables of simple types	Write a program that performs operations and generates output.
10. Programming statements for input	Write a program that accepts keyboard input, performs operations and displays the results.
11. Subprograms; scope; parameters for communication between program parts	Write a program that requires the same code to be executed several times and break it up into units. For example: Distribute the words to a favorite song that contains at least two verses and a chorus. Write one subroutine for each verse and the chorus. Call the subroutines in order. Modify the words of the song with parameters.
12. Structured programming: sequence, selection, repetition	Use each structure in a program as it is introduced.
13. Array variables and/or other aggregate data type	Write a program that utilizes an array or other aggregate data type such as a list, set, etc., depending upon the language employed.
14. Object-oriented programming	Discuss advantages of object-oriented programming. Write a simple program using a real-life object (e.g., Rover is an object of type *dog;* properties: *color* and *breed*; methods: *speak* and *beg*.
15. Program results	Predict the output, given a program and sample input.
16. Comparing high-level languages	Present pieces of code in various languages that accomplish similar tasks (e.g., loop, conditional).

Topic 12: Web Page Design and Development

Topic Description:

The Web Page Design and Development topic will expose students to the steps needed to create simple Web pages. Students will learn to plan and code their Web pages and check for usability.

Textbooks and Supplies:

Computer with Internet access; ability to access a text editor; multiple browsers helpful; space on a web server for publication of completed Web pages recommended.

Time to Complete: 2-3 weeks

Student Learning Objectives	Assessment Measures
The student will be able to:	
1. Correctly use HTML tags to create Web pages.	Written activity Lab activity
2. Apply styles to HTML documents to control presentation.	Written activity Lab activity
3. Express the design of a Web site using standard tools.	Lab activity
4. Create a Web site given design specifications.	Lab activity
5. Publish a Web site.	Lab activity
6. Apply good design techniques when creating a Web site.	Lab activity
7. Explain the difference between interactive and static Web sites	Written activity

Assessment Recommendations: An average of 60% from combined assessment measures is required to demonstrate proficiency in course material.	
Lab activities	70%
Written activities, including exams, quizzes, written assignments	30%

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
1. Terminology	Identify and define key terms associated with Web page development.
2. What makes a good Web site?	Visit various Web sites and discuss good and bad qualities. A discussion should include ease of navigation, clear and concise message, intended audience, and accessibility.

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
3. Design a Web site	Choose a topic of common interest. Brainstorm the content as a class activity. Design the home page. Discuss accessibility. Redesign the home page. Each student selects one content area to design that must represent at least three related pages (e.g., dogs – dog shows, breeds, dogs as pets). A site map and storyboards are created.
4. Discuss HTML tags for these elements: forms, text, graphics, hyperlinks, multimedia, tables	Use each element in an HTML document as it is explained. Check to see that the elements are added in a manner that conforms to World Wide Web Consortium recommendations. Consider factors that affect flexibility such as relative vs. absolute links and server-side vs. client-side scripts.
5. Styles for presentation vs. presentation within markup	Discuss the role of styles vs. the structure and content of a document. Apply various styles to the HTML elements as they are explained. Check to see that the styles are added in a manner that conforms to World Wide Web Consortium recommendations.
6. Create a Web site from the design	Develop all pages. Check to see that the pages conform to World Wide Web Consortium recommendations.
7. Publish a Web site	Upload the pages and visit the site to test it.
8. Evaluate the site	View the site in different ways – multiple browsers, resolution settings, etc. Discussion: Is this a good Web site? Make sure to include accessibility issues in discussion.
9. Interactivity	Introduce the necessity for interactive Web pages. Discuss the dynamic capabilities of the Web and the need for scripting. Visit a dynamic Web site. How does the information that you provide affect what you see? Select a static Web site. List ways to make the site interactive. Will the changes be advantageous to all viewers?
10. Web development tools	Create a Web site using a Web development tool. View the HTML and discuss the advantages and disadvantages of WYSIWYG software.

Topic 13: Multimedia

Topic Description:

This multimedia topic will introduce basic file formats for audio, video, and graphics, and creation of multimedia objects. Appropriate use of multimedia is covered.

Textbooks and Supplies:

Any graphical editing program; ability to play audio/video; microphone(s); Internet connection; word processing and presentation software, and graphical, audio, and video editing freeware recommended; digital still and video camera and scanner if possible.

Time to Complete: 2-3 weeks

Student Learning Objectives	Assessment Measures
The student will be able to:	
1. Explain the differences, advantages and disadvantages between vector and bit-mapped images.	Written activity
2. Based upon the file extension, determine if a given file type is audio, video, or an image and then select the correct tool for viewing the file.	Written activity Lab Activity
3. State the difference between current image formats regarding accessibility and size.	Written activity
4. Convert between image formats.	Lab activity
5. Display a multimedia object within a Web page or document.	Lab activity
6. Determine if a given multimedia object can legally be duplicated and/or distributed.	Written activity

Assessment Recommendations: An average of 60% from combined assessment measures is required to demonstrate proficiency in course material.	
Lab activities	70%
Written activities, including tests, quizzes, and written assignments	30%

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
1. Terminology	Identify and define key terms associated with the use of multimedia objects.
2. Hardware to support multimedia	Research computer hardware necessary to support multimedia – sound cards, video and digital capture cards, digital cameras, scanners, web-cams, digital video camcorders; processor speed; RAM.

Detailed Outline

Focus	Sample Lab / Hands-on Activity
3. Software to support multimedia, including plug-ins for specific media	Research computer software titles that support multimedia, including shareware and freeware; create a list of the media types that each supports. View a Web page without the appropriate plug-in. Download and install the plug-in. Visit the Web page again.
4. Digital imaging	Use a digital camera, microphone, web-cam, digital video camcorder, scanner, etc. if available. Insert the multimedia objects into documents when appropriate using word processing and/or presentation software, or by adding the object into a Web page using HTML.
5. Create, edit, and save bit-mapped and vector images	Create a simple image in any graphical editor (e.g., *Paint*) to become familiar with the various tools. Draw the same simple picture using vector-graphics (e.g., Drawing tools in *Word*).
6. Bit-mapped representation of images, and resolution	Show an original photograph using gradations from very fine to very coarse resolution. Discuss quality vs. amount of information to be stored for uncompressed bit-map. Hand-digitize an image at various (coarse) resolutions using graph paper, and compare storage requirement vs. quality. Scan a document at lowest and highest resolutions. Calculate memory requirements of images at various resolutions taking into account the number of pixels and color depth.
7. Vector vs. bit-mapped images	Place both a bit-mapped image and a vector-based image within a document. Stretch the graphics and compare the results.
8. Image file types and compression	Students compress and uncompress objects like a soft sponge and a balloon to observe that the original object remains unchanged when compressed and decompressed even though the method of compression may differ. Point out that the file extensions for images represent different methods of compression. Use a graphical editor to save the same image in several formats. Evaluate the changes in file size and image quality.
9. Accessibility issues	View Web pages with images suppressed to open a discussion of accessibility for the visually impaired. Expand the discussion to consider the impact of multimedia decisions for individuals with other disabilities. Use accessibility features build into the operating system.
10. Audio and video file types	Discuss the process by which sound and video become digitized. Reinforce the tradeoff between quality and file size. Visit a Web site that contains multiple versions of an audio file with different qualities or record video or sound files. Play each file and compare the quality and file size. Provide a demonstration of working with sound/video if equipment is available.

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
11. Intellectual property rights; ethical issues surrounding the use of images, sound, and video files	Read together some Terms of Use Agreements and discuss appropriate use. Discuss how the editing process affects “truth” (in images, audio and video). If appropriate software is available, modify a graphic in an inappropriate manner (e.g., use the head of one student and the body of another).

Topic 14: Applications

Topic Description:

Students will be required to think analytically to select appropriate tools to solve problems. The problems will require use of major application types such as word processing, spreadsheets, databases, and presentation software. The problems will also require analytical thinking in design and layout of the solution. New concepts in the use of productivity software will be introduced such as macros, multi-table database design, creation, and maintenance, integration between applications, and importing/exporting data.

Textbooks and Supplies:

Appropriate application software.

Time to Complete: 2 weeks

Student Learning Objectives	Assessment Measures
The student will be able to:	
1. Select the appropriate application(s) to use for a particular project.	Written activity Lab activity
2. Use integrated software productively.	Lab activity
3. Design, create, use, and maintain a multi-table relational database	Lab activity

Assessment Recommendations: An average of 60% from combined assessment measures is required to demonstrate proficiency in course material.	
Lab activities and projects	80%
Written activities, including tests, quizzes, and written assignments	20%

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
1. Terminology	Explain terminology as needed throughout unit (e.g., import, export, relation, etc.)
2. Software categories	Have students brainstorm various applications and broadly categorize them (ex: productivity, financial, multimedia, imaging, educational, entertainment, communication). Discuss the purpose of various types of software and how they improve certain tasks. List the strengths and weaknesses of each type.
3. Spreadsheet as a table	Students create a small database using a spreadsheet. Examine the benefits and limitations of using a spreadsheet for database purposes.
4. Relational database design	Students are given a situation, engage in needs analysis, and design a relational database, considering the relationships between tables, data types, and normalization.

Detailed Outline	
Focus	**Sample Lab / Hands-on Activity**
5. Creating and maintaining a database	Students create, modify, and use a multi-table relational database. The implementation must match the design.
6. Retrieving information from a database: queries and SQL	Students use queries to ask specific questions of the database. They revise queries where necessary to narrow their search. The importance of using a structured method of finding information should be stressed.
7. Organizing Information	Students create reports and presentations using various applications, presenting the data in a manner that coherently provides a solution to a stated problem.
8. Problem solving	Students are presented with a problem. They select appropriate software applications and use the selected software to solve the stated problem.
9. Integration	Students are given a file in a text or delimited format and must import it into an application. The data is manipulated and exported to one or more additional applications.
10. Productivity	Students are asked to perform a repetitive process in a given application. A macro is recorded and used to perform the repetitive function. Discuss the benefits of using macros. Are there any limitations? Include security issues in your discussion.

References Used to Create these Standards:

Eide, Arvid, et al. *Introduction to Engineering Design and Problem Solving*. 2nd ed. McGraw-Hill, 2001.

"Funny Headlines." *Quotez*. <http://www.quotations.co.uk>

G. Polya, *How to Solve It*, 2nd ed. Princeton University Press, 1957.

Gottleber, Timothy T., and Timothy N. Trainor. *Even More Excellent HTML with XML, XHTML, and JavaScript*. McGraw-Hill, 2003.

ITEA. *Standards for Technological Literacy: Content for the Study of Technology.* International Technology Educational Association, 2000.

Knowlton, Todd. *C++ Basics*, South-Western Computer Education, 2001.

Litvin, Maria, and Gary Litvin. *C++ for You++: An Introduction to Programming and Computer Science*. Skylight Publishing, 1998.

Logo Foundation. 2003. <http://www.logofoundation.org>

Norton, Peter. *Introduction to Computers.* 5th ed. Glencoe/McGraw-Hill, 2003.

Snyder, Lawrence. *Fluency with Information Technology: Skills, Concepts, & Capabilities*. Pearson Education, 2004.

Squeakland. <http://www.squeakland.org>

Acknowledgements:

Dan Frost, Donald Bren School of Information and Computer Sciences, University of California, Irvine

Bill Madden, Assistant Professor, Bergen Community College, NJ

Walter M. Mayes, Librarian, The Girls' Middle School, CA

Feedback[§] towards improvement of this document was provided by:

Attendees at CS & IT Symposium, June 2005, "Developing Resources to support a National Computer Science Curriculum for K-12"

Attendees at SIGCSE 2005 "Developing Resources to support a National Computer Science Curriculum for K-12"

Ron Cudworth, Monroe Career & Technical Institute, PA

Dan Frost, University of California, Irvine

Barbara Ericson, Georgia Institute of Technology

Duke Mossman, Utah State Office of Education

Yong Cheul Yeum, Korea University

[§] Feedback does not necessarily indicate endorsement.

Appendix A

Level 2 Curriculum: Facets of Computer Science

Computer science can be seen as comprising four facets:

- Computer Hardware
- Using Computer Software
- Solving Problems by Developing Software
- Computers, People, and Society

Each of these facets has principal concepts and underlying themes, which are described below.

Computer Hardware

The development of the electronic computer has been one of the technological marvels of the last century. Research and development of computers and peripherals actively continues.

Principal concepts and themes of this facet are:

1. At a fundamental level, all computers are collections of circuits. (Although circuits and electronics are properly covered in high school physics, a brief introduction in the context of CS is appropriate for the Level 2 curriculum.)
2. The most common architecture for computers is based on a central processor, memory, and peripherals.
3. Memory storage devices (including punch cards, paper tape, cassette tapes, hard disk drives, floppy disks, CD-ROMs and DVDs, memory sticks, RAM, ROM, cache, and video memory) have a variety of characteristics.
4. Memory storage devices usually store information in units called bytes, and each byte has a numeric address.
5. Computer processors (chips) are almost ubiquitous in cars, cell phones, traffic signal controllers, and other embedded devices.
6. Peripherals serve two main functions, input and output. Specialized peripherals are used in non-PC devices such as robots, satellites, cell phones, digital camera.
7. Computers are connected in networks via a wide variety of communication media, including telephones (with modems), Ethernet cable, and wireless.
8. Various tools (screwdrivers, Allen wrenches) are used to open computer cases and add or remove components.
9. Safety concerns must be kept in mind when assembling or fixing computers or any electronic equipment.

Using Computer Software

Some software is written to be tightly integrated with specific hardware, as in a cell phone or a digital camera. In other cases, a software application, such as a word processor or a Web browser, is primarily designed to run on standard hardware. Sometimes software works "behind the scenes" and can be almost invisible, as in the Internet or many parts of an operating system. Familiarity with a variety of computer software programs and with the basic concepts underlying many of them is a prerequisite for many jobs and for understanding a large part of 21st century culture.

Principal concepts of this facet are:

1. Software may be tightly or loosely coupled with the computer hardware on which it runs.
2. By representing information in digital format, computers can store, manipulate, and transmit that information.
3. The instructions a computer follows are in software, which means that they can be changed easily.
4. Related data is often stored in a "file." A file (usually) corresponds to a particular range of locations in a memory storage device. The data in the file often has a specific format. A file has a name, and often the name has an extension that is associated with a specific program that knows how to use the contents of the file.
5. Files are organized in a hierarchy of folders or directories.
6. Files should be backed up periodically.
7. The same data can often be displayed in multiple ways.
8. Personal computers' operating systems often have graphical user interfaces.
9. Commercial "shrink-wrapped" software includes categories such as word processors, Web browsers, presentation and slide managers, spreadsheets, databases, graphic and music content creators and editors, and e-mail clients.

Solving Problems by Developing Software

Computer software solutions are created by identifying a need or opportunity, analyzing how it can be addressed with software, designing and coding the program, carefully testing the program, and in many cases writing documentation and training the users. Gaining a basic understanding of how software is created gives students a deeper understanding of what computers can do.

Principal concepts of this facet are:

1. Creating software involves several common phases:
   (a) Identifying the requirements from the user's perspective
   (b) Planning how to write the program (particularly important when the program is large or more than one programmer will collaborate)
   (c) Following the plan by writing code in a computer programming language
   (d) Testing the program to assure it meets the original requirements, runs acceptably quickly, is user-friendly, and has other desired qualities
   (e) Turning the program over to the user, which involves training, documentation, and designing procedures

   Often, steps (d) and (e) are repeated after beta-testing by end users.
2. Computers follow programs, which are written by humans.
3. Computer programs are written in computer languages, which have rigid syntax utilizing a limited number of key words and symbols.
4. Computers have no "understanding" beyond what is explicitly coded into a computer program.
5. For a problem to be solved by a computer, every step must be defined in detail.
6. For information to be used by the computer, it must be represented as digital data.

7. Writing a computer program involves selecting or creating algorithms and data structures, and analyzing their performance and other characteristics. Algorithms to solve a specific problem vary widely, and often involve different trade-offs of space and time.
8. Algorithms implemented in computer programs are made up of elementary control structures, such as conditionals, loops, and subroutines.
9. Computer science has been developed in the 20th and 21st centuries, but the philosophic roots of the "laws of thought" and algorithmic thinking originated with Plato and the pre-Socratics.

Computers, People, and Society

Technical advances have driven social changes throughout history, and tools have shaped culture in many ways. The rapid development of computers, networks, and peripherals has an ongoing impact on society.

Principal concepts of this facet are:
1. Computer technology and software changes more quickly than ethics and laws, thus creating a constant tension in society.
2. The ubiquity of data in digital format presents new issues of privacy and security.
3. Computerized data is often copied and rarely deleted, raising issues of privacy, ethics, and ownership rights.
4. Humans are best at recognition, making connections between similar things, and learning by doing. Computers are best at following small instruction steps and processing digital data quickly and consistently.
5. A human-computer interface is the meeting point of the human and computer realms. A good interface minimizes the human's short-term memory load, is compatible with a diverse set of users, and prevents errors[**].
6. Computers are tools with several functions: to process data (to compute), to store data, to acquire and display data, and to move data from one computer to another (to communicate).

Beginning on the next page, we've included a cross-reference between these facets of computer science and the Level 2 curriculum's topics and focuses.

[**] Shneiderman, Ben. *Designing the User Interface.* Third Ed. Addison Wesley, 1998.

Topic	**Focus**	**Computer Hardware**	**Using Software**	**Developing Solutions**	**People & Society**
1 Principles of Computer Organization	1. Terminology	X	X		
	2. Hardware components	X			
	3. Software components		X		
	4. Interaction of components	X	X		
	5. Purchasing a computer	X			X
	6. File systems		X		
	7. Troubleshoot problems	X	X		
2 Problem Solving	1. Problem-solving process			X	
	2. Understanding the problem			X	
	3. Exploring problems			X	
	4. Problem design			X	
	5. Problem data			X	
	6. Solution accuracy			X	
	7. Program coding and testing			X	
	8. Re-evaluation and refinement			X	
	9. Communicate results			X	X
3 Computer Networks	1. Terminology	X	X		
	2. Data transfer	X	X		
	3. Data collision	X	X		
	4. Connecting a computer	X	X		
	5. Identity	X	X		
	6. Applications of networks	X	X		
	7. Mobile computing	X	X		X
	8. Network Security	X	X		X
4 Internet Concepts	1. Internet elements		X		X
	2. Search engine fundamentals		X		
	3. Search engines and directories		X		
	4. Refining search parameters		X		
	5. Evaluating Web sites		X		X
	6. Security on the Internet		X		X
5 Hierarchy and Abstraction in Computing	1. Decomposing the complex			X	
	2. From source to execution	X		X	
	3. The role of circuits	X			
	4. The power of hierarchy			X	
	5. Abstraction			X	

Topic	**Focus**	**Computer Hardware**	**Using Software**	**Developing Solutions**	**People & Society**
6 Connections between Mathematics and Computer Science	1. Binary number system	X	X	X	
	2. Encryption		X	X	X
	3. Conditional and Boolean logic			X	
	4. Functions			X	
	5. Set representation			X	
7 Models of Intelligent Behavior	1. What is intelligence?				X
	2. Natural language			X	X
	3. Knowledge-based systems		X	X	X
	4. Machine learning		X	X	X
	5. Game playing, searching		X	X	
	6. Robotics		X	X	
	7. Vision and speech		X		X
	8. Myth of intelligent behavior		X	X	X
8 Interdisciplinary Utility of Computers and Problem Solving	1. How are computers used?		X		X
	2. Information storage and retrieval		X		
	3. Decision-making support		X		X
	4. Data visualization		X		X
	5. Communications		X		X
	6. Modeling and design		X	X	
	7. Art, music, video		X	X	
	8. Education and training		X		
	9. E-commerce		X		
	10. Embedded systems	X	X		
	11. More examples	X	X		X
9 Ethical Issues	1. Terminology				X
	2. How technology has changed				X
	3. The effect of technology				X
	4. Privacy and sharing of information				X
	5. Intellectual property				X
	6. Responsible use of software		X		X
10 Careers in Computing	1. Job/Career titles				X
	2. Guest lecturers				X
	3. Current events				X
	4. Job availability				X
	5. The perfect job				X

Topic	**Focus**	**Computer Hardware**	**Using Software**	**Developing Solutions**	**People & Society**
11 Programming Languages	1. Terminology			X	
	2. Representation of text			X	
	3. Representation of numbers			X	
	4. Data types			X	
	5. Program execution			X	
	6. Programming design techniques			X	
	7. Programming style			X	
	8. Programming statements for output			X	
	9. Declaring constants and variables			X	
	10. Programming statements for input			X	
	11. Subprograms; scope			X	
	12. Structured programming			X	
	13. Array variables			X	
	14. Object-oriented programming			X	
	15. Program results			X	
12 Web Page Design and Development	1. Terminology	X	X	X	X
	2. What makes a good Web site?		X		X
	3. Design a Web site		X	X	X
	4. HTML tags			X	
	5. Styles and markup			X	
	6. Create a Web page from a design			X	
	7. Publish a Web site		X	X	
	8. Evaluate the site		X	X	X
	9. Interactivity			X	X
	10. Web development tools		X	X	
13 Multimedia	1. Terminology	X	X	X	
	2. Hardware to support multimedia	X			
	3. Software to support multimedia		X		
	4. Digital imaging	X	X		
	5. Create, edit, and save images		X		
	6. Bit-mapped representation of images		X	X	
	7. Vector vs. bitmapped images		X	X	
	8. Image file types and compression		X		
	9. Accessibility issues		X		X
	10. Audio and video file types		X		
	11. Intellectual property rights				X

Topic	**Focus**	**Computer Hardware**	**Using Software**	**Developing Solutions**	**People & Society**
14 Applications	1. Terminology		X	X	
	2. Strengths and weaknesses		X		X
	3. Spreadsheet as a table		X	X	
	4. Relational database design		X	X	
	5. Creating and maintaining a database		X	X	
	6. Retrieving info from a database		X	X	
	7. Organizing information		X	X	X
	8. Problem solving		X	X	
	9. Integration		X		
	10. Productivity		X	X	X