Association for Computing Machinery
Committee for Computing Education in Community Colleges (CCECC)

Cybersecurity-infused Curricular Guidance for Associate-Degree Transfer Programs in Computer Science

IronDog, **October 2016**
last draft version
http://ccecc.acm.org/ACMCompSciLastDraft/

Available for Public Review and Comment
from October 25 to November 30, 2016

Feedback being collected at
www.surveymonkey.com/r/IronDog-Feedback

Table of Contents

Acknowledgements

The members of the ACM Committee for Computing Education in Community Colleges (CCECC) acknowledges and thanks the ACM Education Board for providing funding for this important project to develop cybersecurity-infused curricular guidance for associate-degree transfer programs in computer science.

ACM CCECC Members

- Dr. Elizabeth K. Hawthorne, Union County College, NJ
- Dr. Cara Tang, Portland Community College, OR
- Prof. Cindy S. Tucker, Bluegrass Community and Technical College, KY
- Dr. Christian Servin, El Paso Community College, TX
- Prof. Teresa T. Moore, Volunteer State Community College, TN

Computer Science-Cybersecurity Team Leaders

- Prof. Lambros Piskopos, Wilbur Wright College, IL
- Dr. Christian Servin, El Paso Community College, TX
- Prof. Teresa T. Moore, Volunteer State Community College, TN

Computer Science-Cybersecurity Task Force Members

- Prof. Kimberly Bertschy, Northwest Arkansas Community College, AR
- Prof. Colleen Case, Schoolcraft College, MI
- Dr. Markus Geissler, Cosumnes River College, CA
- Dr. Becky Grasser, Lakeland Community College, OH
- Prof. Charles Hardnett, Gwinnett Technical College, GA
- Prof. Amardeep Kahlon, Austin Community College District, TX
- Prof. James Kolasa, Bluegrass Community and Technical College, KY
- Dr. Shamsi Moussavi, MassBay Community College, MA
- Prof. Pam Schmelz, Ivy Tech Community College, IN
- Prof. Melissa Stange, Lord Fairfax Community College, VA
- Prof. Khallai Taylor, Miami-Dade College, FL
- Prof. Carole Tharnish, Northeast Community College, NB

Other Contributors

- Dr. Anne Applin, Southern Maine Community College, ME
- Prof. Bryce Barrie, Saskatchewan Polytechnic, Canada
- Prof. Michael Bauer, Leeward Community College, HI
- Prof. Paul Dadosky, Ivy Tech Community College, IN
- Prof. Andrea DeMott, Ohio University, OH
- Dean Jamie Edwards, Wytheville Community College, VA
- Prof. Rafael Escalante, El Paso Community College, TX
- Dr. Larry Forman, San Diego City College, CA
- Prof. Dianne Hill, Jackson College, MI

- Dean Nancy Jones, Coastline Community College, CA
- Prof. Marc Nester, Wytheville Community College, VA
- Dr. Dean Nevins, Santa Barbara City College, CA
- Dr. Michael Posner, Villanova University, PA
- Prof. Kristopher Roberts, Ivy Tech Community College, IN
- Prof. Barry Sullens, Ivy Tech Community College, IN
- Prof. Robert Surton, Columbia Gorge Community College, OR

Introduction

Overview of the Curricular Development Process

The comments from the first public review and comment period for the initial draft, *StrawDog*, have been processed to produce the second and last draft called ***IronDog***. This iterative curriculum development process updates the *Guidelines for Associate-Degree Transfer Curriculum in Computer Science,* published by the ACM Committee for Computing Education in Community College (CCECC) in 2009. To date, over 50 community college and university computing educators contributed to the creation of this version as either members of the CS-Cyber task force who met virtually in teams over the course of months or through participating in a half day workshop at SIGCSE 2016 in Memphis, TN or stopped by the poster session at ITiCSE 2017 in Arequipa, Peru.

We continue to engage the international community by requesting constructive feedback on ***IronDog****, our last draft.* The members of the ACM CCECC thank you for taking the time to provide your insights and comments via **www.surveymonkey.com/r/IronDog-Feedback**.

The professional societies of the ACM and the IEEE Computer Society have a long history of collaborating on computing materials for higher education. These organizations have jointly produced significant volumes of curricular recommendations and guidelines for associate, baccalaureate and graduate computing programs; these volumes are referred to as the ACM Computing Curricula series (www.acm.org/education/curricula-recommendations). Likewise, the ACM Committee for Computing Education in Community Colleges (CCECC) has produced a corresponding set of curricular guidelines that provide similar guidance for associate-degree granting institutions, in a manner that fosters inter-institutional cooperation and student articulation. This model curriculum provides discussion on transfer considerations and discussion on articulation in computer science.

Like most countries, cybersecurity is a national priority in the United States with a formal action plan (whitehouse.gov/the-press-office/2016/02/09/fact-sheet-cybersecurity-national-action-plan). Higher education is a key component of the cybersecurity national action plan (NCAP). ACM responded by integrating information assurance and security learning outcomes throughout its most current computer science curricular guidance, CS2013. In 2015, ACM in association with the Computer Society of the Institute of Electrical and Electronics Engineerings (IEEE-CS), the Association for Information Systems (AIS), the Cyber Education Project (CEP), and the International Federation of Information Processing (IFIP) formed a joint task force to create undergraduate curricular guidelines in cybersecurity (www.csec2017.org). In similar fashion and informed by the *National Cybersecurity Workforce Framework* (csrc.nist.gov/nice/framework/), the ACM CCECC is integrating contemporary cybersecurity content into the revision of its 2009 *Guidelines for Associate-Degree Transfer Curriculum in Computer Science*.

Survey Input

Two surveys provided valuable input influencing these curricular guidelines. The first survey asked which knowledge areas and knowledge units from CS2013 are appropriate in the first two years of a computer science program. A second survey solicited input on cybersecurity content

appropriate in a computer science program in the first two years. The surveys were disseminated internationally and received approximately 50 responses from eight different countries. Responses to the first survey indicated that 16 of the 18 knowledge areas of CS2013 are appropriate, in some part and at some level, in the first two years of a computer science transfer program.

Two-Year/Community College Environment

According to the American Association of Community Colleges, nearly one-half of all undergraduates in the United States are enrolled in two-year colleges, and more than half of all first-time college freshman attend community and technical colleges. "Community colleges are centers of educational opportunity. They are an American invention that put publicly funded higher education at close-to-home facilities, beginning nearly 100 years ago with Joliet Junior College (in Joliet, Illinois). Since then, they have been inclusive institutions that welcome all who desire to learn, regardless of wealth, heritage, or previous academic experience. The process of making higher education available to the maximum number of people continues to evolve" (www.aacc.nche.edu/).

The community college environment is uniquely positioned, resulting from the threefold mission of these institutions to provide a learning environment for:

1) transfer into baccalaureate programs;
2) entrance into the local workforce; and
3) lifelong learning for personal and professional enrichment.

In addition, many two-year colleges are drivers of local economic development, providing workforce development and skills training, as well as offering noncredit programs ranging from English as a second language to skills retraining to community enrichment programs or cultural activities.

Two-year colleges serve high school graduates proceeding directly into college, workers needing to upgrade skill sets or master new ones in order to re-enter the workforce, immigrants seeking to become integrated into the local culture and master a new language, individuals leaving the workplace to engage college-level coursework for the first time, returning students with college degrees who have decided to pursue an alternate career path, and many individuals in need of ongoing training and skill updating. This diversity is addressed in numerous ways, including targeted career counseling, remediation of basic skills, specialized course offerings, individualized instruction and attention, flexible scheduling and delivery methodologies, and a strong emphasis on retention and successful completion. Furthermore, because two-year colleges have less restrictive entrance requirements, faculty must be prepared to instruct students exhibiting a broad range of academic preparations, aptitudes, and learning styles. The mission of two-year college faculty is to focus their full-time attention on effective pedagogy for educating a diverse student population, remaining current in their discipline and in the scholarship of teaching and learning, and fostering student success. Two-year, community or technical colleges, as well as certain four-year colleges, award associate degrees to students completing between 60 to 66 higher education credits in a

particular program of study. It is often the case that an associate-degree requires approximately half the college credit of a bachelor's degree. Associate-degree programs are complete in their own right, whether designed specifically to enable graduates to transfer into the upper division of a baccalaureate program or to gain entry into the workforce. These institutions also offer certificate programs, intended to be fulfilled in less time than a complete degree program; such programs are often designed for targeted student audiences and focused on specific content.

At the earliest opportunity, faculty and academic advisors must help each student determine which type of program best serves the student's educational and career goals. Such considerations include the distinctions between certificate, career and transfer programs, the academic requirements of each, and the associated employment options. Career-oriented associate-degree (typically AAS) programs provide the specific knowledge, skills, and abilities (KSA) necessary to proceed directly into the workplace, while transfer-oriented degree (typically AS) programs provide the academic foundation and pathway to continue a program of study at a four-year college or university.

Diversity in the Computing Profession

Across the globe there is a high demand for computing professionals and a significant shortfall in satisfying job vacancies in many locations. In the U.S. alone, it is anticipated that the current graduation rates in computing disciplines may satisfy only one-third of the projected 1.4 million computing-related jobs openings in the coming years. Worldwide, the growth of new and emerging roles in computer, technology, and engineering fields exceeds the rate that underrepresented groups enter these fields. Academic research continues to bear light on the pressing need to increase the diversity of students pursuing computer science degrees and the numerous benefits of doing so. To help fulfill the increasing shortage of computer professionals, computer science faculty should increase efforts to effectively recruit and retain a wider range of student and build and provide effective support structures so that all students can successfully graduate.

Cybersecurity in Computing Curricula

Whether referred to as "cybersecurity", "computer security", "information security", "information assurance" or some other name, curriculum content in creating and maintaining secure computing environments is a critical component in associate-degree computing programs. Almost every career path open to a computing student encompasses some aspect of security. System administrators and engineers must be able to properly design, configure and maintain a secure system; programmers and application developers must know how to design and build secure, fault-tolerant software systems from the bottom up; web specialists must be capable of assessing risks and determining how best to reduce the potential impact of breached systems; user support technicians must be knowledgeable in security concerns surrounding desktop computing; and project managers must be able to calculate the cost/benefit tradeoffs involved with implementing secure systems.

It is the responsibility of faculty to ensure that students are well prepared for the cybersecurity challenges they will inevitably encounter in their careers as computing professionals. This can be addressed by way of a variety of implementation strategies. One approach that some

associate-degree computing programs offer is a host of individual courses on specific security topics. This approach can provide a wealth of content opportunities for specialization but may create scheduling challenges for many students as it runs the risk of students graduating without having taken sufficient electives to achieve the understanding of the security concepts necessary to function in their professional roles.

Another approach is to fully integrate and incorporate contemporary cybersecurity content into core, introductory computer science courses with specialized courses reserved for targeted settings. These guidelines employ the integrated approach, incorporating relevant cybersecurity student learning outcomes throughout the computer science body of knowledge for lower division, undergraduate curriculum. The Committee also strongly advocates for learning activities that require students to actively demonstrate mastery of the tenets of professional conduct, ethical and responsible behavior, as well as an appreciation for cybersecurity in a holistic manner.

Ethics and Professionalism

Ethical reasoning and professional conduct are important concepts in the overall curricula for computing disciplines including computer science, and must be integrated throughout the programs of study. This ethical and professional context should be established at the onset and should appear routinely in discussions and learning activities throughout the curriculum. The ACM Code of Ethics notes that "When designing or implementing systems, computing professionals must attempt to ensure that the products of their efforts will be used in socially responsible ways, will meet social needs, and will avoid harmful effects to health and welfare." The Code goes on to provide an excellent framework for conduct that should be fostered beginning early in students' experiences. *(www.acm.org/about-acm/acm-code-of-ethics-and-professional-conduct)*.

As computing technologies become ubiquitous in society, ethical behavior and adherence to codes of conduct for computing professionals is imperative; therefore, careful consideration of legal, ethical, and societal issues involving computing, the Internet and databases are essential to the education of computing professionals. Students who realize the potential uses and abuses of technology will, as citizens, be able to contribute to public policy debate from a knowledgeable perspective on issues such as property rights and privacy concerns that affect everyone.

Computer systems have substantial social impact in nearly every setting including applications such as healthcare, finance, transportation, defense, government, education, and communications; real-time and safety-critical systems typically have acceptable margins of error close to nil. Developers and support technologists of such computing systems are confronted by challenges regarding choices and tradeoffs in the design, implementation, and maintenance of these systems. Engaging students in the consideration of the ethical aspects for such decisions as well as giving them practice in identifying and weighing the ethical issues enables them to make more judicious choices. It is crucial that students pursuing computing careers be made aware of and properly equipped to handle the complexities of professional judgments; as

computing professionals, graduates must follow codes of conduct and take responsibility for their actions and be accountable for the systems that they develop and support.

Characteristics of Computer Science Graduates

Graduates of associate-degree programs in computer science should possess foundational competency in the areas described throughout the Body of Knowledge (BoK) presented in these curriculum guidelines. Increasingly, the area of computing has become critical to the operation of many organizations. Computing employees must demonstrate professionalism and ethical behavior (as described above), adhere to codes of conduct, safeguard confidentiality, and respect privacy. They must take responsibility for their actions, be accountable to the organization, understand the impact of their work on others, and demonstrate effective and efficient work practices. This field also demands that professionals engage in lifelong learning, an ongoing process of professional growth and development, to ensure that their skills and abilities remain current with ever-changing technology.

To this workplace readiness goal, faculty are strongly encouraged to incorporate professional practices and applied work as an integral part of computer science programs to benefit students. Students should be encouraged to:

- work in teams;
- use techniques of task and time management;
- solve practical problems in course projects;
- make oral presentations;
- communicate effectively in writing;
- confront issues of privacy, confidentiality and ethics;
- use current technology in computer laboratories;
- attain real-world experience through cooperative education, internships, and/or other practicum activities; and
- participate in student chapters of computing societies and organizations (e.g., ACM) for professional development opportunities.

Internationalization

In the process of developing its curricular guidance in computer science, the ACM CCECC continually seeks international perspectives from two-year post-secondary (“tertiary”) higher education programs both locally within the United States and globally throughout the world. Feedback on this early draft is being sought from countries around the globe, such as China, India, Turkey, South Africa, Australia, New Zealand, Canada, Mexico, Peru, Brazil, the United Kingdom, and countries across the European Union.

Furthermore, the American Association of Community Colleges (AACC) has established an Office of International Programs and Services whose stated goals are “to advocate the community college role in global education among key constituencies, nationally and internationally, to advance global exchanges and partnerships between member colleges and international entities, and to promote intercultural understanding and engagement among students, faculty, staff, and decision makers.” James McKenney, AACC Vice-President for Economic Development and International Programs, spoke as early as 2002 in a reflective

interview titled “The Global Linkage” appearing in the journal “US Society and Values” about the rapidly expanding phenomenon of “community colleges”, “two-year technical colleges”, and the “two-year structure” in general throughout Europe, the Americas and Asia.

This phenomenon has since been reported on, promoted and codified in any number of publications and resources. For example, the Council for Industry and Higher Education (CIHE) and The Mixed Economy Group in England have prepared a comprehensive report titled “Higher Education and Colleges: A Comparison Between England and the USA”. The United Nations-affiliated Institute of International Education provides a wealth of resources regarding post-secondary and higher education around the world, including activities of institutions akin to the two-year colleges of the United States. The Paris-based Organization for Economic Cooperation and Development (OECD) provides background information, analyses and recommendations for “opportunities for education in the years after compulsory schooling” across Europe, as does the Directorate-General for Education and Culture of the European Commission. The “University World News” and the “World Education News and Reviews” publications are sources of current events worldwide in post-secondary education; the Community College Research Center at Columbia University also provides links to such documentation. The Association of Canadian Community Colleges (ACCC) offers a wealth of information on two-year colleges in Canada. The Ministry of Education of the People’s Republic of China describes institutions similar to community colleges in its discussion on “2 to 3-year higher vocational education with the emphasis on high-level professional technical talents.”

Assessment

The cognitive outcomes in this guidance are clustered by related Knowledge Areas (KAs). The KAs are further organized into Knowledge Units (KUs) with a list of corresponding student learning outcomes. Each learning outcome is accompanied by an assessment rubric. The ACM CCECC uses a structured assessment metric comprised of three tiers: “emerging”, “developed”, and “highly developed.” Typically as the level of student achievement progresses from “emerging” to “highly developed”, the level of Bloom’s verbs also increases from the lower order thinking skills (LOTS) to the higher order thinking skills (HOTS). It is important to note that the middle tier of “developed” indicates the student has achieved the intended level of the learning outcome.

Articulation

Articulation is a key consideration in associate-degree programs which are designed as transfer curricula. Articulation of courses and programs between academic institutions is a process that facilitates transfer by students from one institution to another. The goal is to enable students to transfer in as seamless a manner as possible. Efficient and effective articulation requires accurate assessment of courses and programs as well as meaningful communication and cooperation. Both students and faculty have responsibilities and obligations for successful articulation. Ultimately, students are best served when educational institutions establish well defined articulation agreements that actively promote transfer.

Articulation agreements often guide curriculum content as well, and are important considerations in the formulation of transfer-oriented programs of study. Institutions are

encouraged to work collaboratively to design compatible and consistent programs of study that enable students to transfer, in the United States from associate-degree programs into baccalaureate-degree programs, and in other countries from post-secondary colleges into universities. A two-year college must develop transition and articulation strategies for the colleges and universities to which its students most often transfer, recognizing that it may be necessary to modify course content to facilitate transfer credit and articulation agreements. A program of study must also take into consideration the general education requirements at both the initial college and the anticipated transfer institution. Faculty must ensure that they clearly define program goals, address program learning outcomes, and evaluate students effectively against defined course outcomes. Articulation agreements should specify one or more well-defined exit points for students to matriculate from the post-secondary college to the transfer institution. In turn, faculty at the receiving institution must provide any transitional preparation necessary to enable transfer students to continue their academic work on par with students at their institution. Hence, students must expect to complete programs in their entirety up to well-defined exit points (e.g., completion of a defined course sequence or program) at one institution before transferring to another institution; one cannot expect articulation to accommodate potential transfers in the middle of a carefully designed curriculum. Acting on these considerations, all post-secondary institutions of higher education will foster student success and best serve their students' academic and career aspirations.

Transfer Programs

Typically associate-degree computing programs fall into two categories: those designed for transfer into baccalaureate-degree programs (Associate in Science) and those designed to prepare graduates for immediate entry into career paths (Associate in Applied Science). Colleges should make students aware at the onset of their studies of the distinctions between career and transfer programs, the academic requirements of each, and the resultant employment options. Transfer-oriented associate-degree programs rely on formal inter-institutional articulation agreements to ensure that students experience a seamless transition between lower division associate-degree coursework and upper division baccalaureate-degree coursework. Articulation of courses and programs between two academic institutions facilitates the transfer of students from one institution to the other. Faculty and students alike have responsibilities and obligations to achieve successful articulation.

Efficient and effective articulation requires a close evaluation of well-defined course and program outcomes as well as meaningful communication and cooperation. For example, a particular course in one institution might not be equivalent to a single course at a second institution; however, a group or sequence of courses could be determined equivalent to another course grouping or sequence. Faculty must ensure that they clearly define program requirements, address program goals in a responsible manner, and assess students effectively against defined standards. When specifying points of exit within the articulation agreement document, faculty at the transferring institution must provide sufficient material to prepare students to pursue further academic work at least as well as students at the second institution.

It is not uncommon for students to complete an associate-degree program of study, choose to work for a period of time, and then return to college to pursue their upper division studies for

career advancement. (And many employers will provide tuition reimbursement for workers who wish to continue toward a baccalaureate degree.) Because of the ever evolving nature of computing, students must be aware that course content and program requirements are updated frequently, potentially subjecting them to new program requirements and revised articulation agreements. Students are best served when sequences of courses are completed as a unit at one institution due to the comprehensive and conceptual nature of the computing and mathematics content. Hence, students should complete programs of study in their entirety up to well-defined exit points at one institution before transferring to another institution; articulation cannot be expected to accommodate potential transfers in the middle of a well-defined and recognized body of knowledge.

Academic institutions are advised to work collaboratively to design compatible and consistent programs of study that enable students to transfer easily from associate-degree programs into baccalaureate-degree programs. In support of this goal, the ACM provides curricular guidelines for both associate- and baccalaureate-degree programs in computer science.

Career Programs

Typically associate-degree programs fall into two categories: those designed to prepare graduates for immediate entry into career paths and those designed for transfer into baccalaureate-degree programs. Colleges should make students aware at the beginning of their studies of the distinctions between career and transfer programs, the academic requirements of each, and the resultant employment options. Students graduating from a career-oriented associate-degree computing program will typically enter the workforce directly upon graduation.

Career-oriented associate-degree programs provide students with the specific knowledge, skills and abilities (KSA) necessary to proceed directly into employment in a targeted work environment. The program of study will include professional development coursework as well as courses that emphasize communication skills, mathematical reasoning and other general education requirements. The degree granted upon completion of a career-oriented program is typically an Associate in Applied Science (AAS). In addition, many students will augment their formal studies with technical certifications to enhance their immediate employability.

The following factors support the viability of a career-oriented associate-degree program and help ensure the success of students in the workplace:

- An active industry advisory committee consisting of prospective employers, providing guidance concerning the knowledge, skills, and abilities students must possess to enter directly into a career within their community.
- Real-world work experience including co-op programs, internships and other practicum activities, with an emphasis on professional practices. Core and elective coursework as recommended by advisory committees.
- Integration of technical, communication and time-management skills, team projects, and other interpersonal skills that prepare the student for a business working environment.
- Potential articulation paths that enable the career-oriented student to pursue a baccalaureate degree in the future after working for some period of time.

- Assessment processes whereby students can earn credit for relevant experience.

It is important to note that a career-oriented associate-degree program is not intended to facilitate transfer into a baccalaureate program, but rather to provide entry into a career that requires specialized post-secondary skills and an advanced level of expertise and education. Nevertheless, many students graduating from career-oriented programs subsequently select to further their education at the baccalaureate level (frequently with employer tuition assistance plans).

Computer Science Laboratory Experience

The computer laboratory experience is an essential part of the computing curriculum, either as an integral part of a course or as a separate stand-alone course. Such experiences should start early in the curriculum, the very beginning when students are often motivated by the "hands-on" nature of computing. Introductory laboratories should be designed and conducted to reinforce concepts presented in lecture classes and homework. Students should be provided many opportunities to observe, explore and manipulate characteristics and behaviors of actual devices, systems, and processes. Every effort should be made by instructors to create excitement, interest and sustained enthusiasm in computing students. Many associate-degree granting institutions will be familiar with strong lab-based learning activities, drawing on years of experience with programs such as electronics technology and industry-provided networking curricula. Numerous colleges have long recognized that experiences such as survey courses in engineering often engage students in stimulating activities that peak their interests and set the stage for career choices in such fields. These colleges will find that they can leverage existing facilities, resources and faculty expertise in implementing computing programs.

Mathematics Requirements

A strong foundation in mathematics provides the necessary basis for associate-degree transfer programs in computing. This foundation must include both mathematical techniques and formal mathematical reasoning. Mathematics provides a language for working with ideas relevant to computing, specific tools for analysis and verification, and a theoretical framework for understanding important concepts. For these reasons, mathematics content must be initiated early in the student's academic career, reinforced frequently, and integrated into the student's entire course of study. Curriculum content, pre- and co-requisite structures, and learning activities and laboratory assignments must be designed to reflect and support this framework. Many students enter two-year colleges with insufficient mathematics preparation for a computing program. Such students must devote additional semesters to achieve the mathematical maturity and problem-solving skills required to be successful in computing coursework.

The concepts established in a course on **Discrete Structures** are foundational material for computer science, and for that reason such coursework must be completed early in the program of study. A typical Discrete Structures course includes requisite concepts in set theory, induction, recursion, logic, graph theory, and combinatorics, and uses the notion of formal mathematical proof as a unifying theme. These concepts are critical to the study of data structures and algorithms. This foundational course can be taught very successfully by

computer science faculty with appropriate qualifications; in this manner, the content can be presented from the computing perspective, with examples and assessment activities tailored to that perspective as well. Concepts in discrete structures also serve as underpinnings for advanced computer science topics. For example, an ability to create and understand a formal proof is essential in formal specification, in verification, and in cryptography; professionals use graph theory concepts in networks, operating systems, and compilers and set theory concepts in software engineering and in databases.

The theoretical concepts of **Calculus** are required for the study of efficiency of algorithms and measured by Big-O notation. An introductory Calculus course includes the foundational concepts of limits, functions, and upper and lower bounds necessary for understanding asymptotic analysis. Mathematics faculty typically teach this course, intended for engineering, science or mathematics majors. For computer science majors, the ability to think abstractly and to generate software solutions of mathematical models for real-world scenarios is enhanced through the study of calculus.

Laboratory Science Requirements

Rigorous laboratory science courses such as physics, chemistry and biology provide computer science students with direct hands-on laboratory experiences and strong training in the tenets of the scientific method. The scientific method presents a structured methodology for much of the discipline of computing; it also provides a process of abstraction that is vital to developing a framework for logical thought. Learning activities and laboratory assignments found in computer science courses should be designed to incorporate and reinforce this framework of experimentation and observation. Furthermore, advisors should guide students intending to transfer into a baccalaureate program (immediately or as a long-term goal) to select specific science coursework appropriate to that objective. Program requirements of this nature can provide students with a crucial foundation should they later pursue interdisciplinary computing careers in scientific domains, such as astronomy, geography, biology, chemistry, or physics.

Student Support Services

The Student Support Services provide assistance to students particularly through counseling, mentoring, and tutoring. The format of these services may vary depending on the institution's mission and/or the academic program. Frequently, student support services are comprised of enrolled students and student managers from the same institution, supervised by professional staff/faculty members. These occupation opportunities support students with an opportunity to help their peers in addition to obtain professional experience while continuing with their studies. Student support services are primarily focus on student success and retention. Active components in these supportive services focus on retention programs. Examples of these programs may occur through:

- Tutoring Learning Centers*:* provide academic support to students through tutoring and computer assisted instruction for various subjects. TLCs are intended to enhance fundamental learning skills such as teamwork, communication, and negotiation skills. These centers provide a supportive environment to promote positive assistance to students as well as to provide supplemental resources to the course work.

- Per-led Team Learning Models*:* provide active and collaborative discussion between students in the introductory courses. Usually these sessions take place in a peer-based learning environment, whether in an open lab or round tables at a learning center facility. These sessions allow peer leaders to elaborate concepts through active learning strategies to questions and struggles students are likely facing.
- Counseling Centers*:* provide assistance in career pathways to students who are undecided in which track of computing to specialize. These centers provide options in student's computing based, interest and current academic background, including English skills, ESL, and Math. Normally, counselors work hand-by-hand with faculty in the corresponding programs assisting student and providing available options in academic programs such as associate and certificates of completion.

Mapping to the Body of Knowledge

Once these guidelines are finalized, the ACM CCECC will provide an online process (web form) for mapping your computer science program to the computer science transfer Body of Knowledge. If you are interested in receiving an email notification when the mapping process is ready, please contact the Committee at http://ccecc.acm.org/contact and indicate CS transfer mapping.

The Body of Knowledge with Assessment Metrics

The ACM/IEEE CS2013 Body of Knowledge (BoK) serves as the foundational curricular framework for these associate-degree transfer guidelines in computer science. The CS2013 BoK is organized into a set of 18 Knowledge Areas (KA) that correspond to topical areas of study in computing for undergraduate, baccalaureate degree programs in computer science. These associate-degree transfer guidelines include 16 of the 18 Knowledge Areas, purposefully excluding the Intelligent Systems and Platform-based Development KAs, each for different reasons. The Intelligent Systems KA consists mostly of elective content that is more appropriate for upper division undergraduate instruction. There are no hours associated with the Platform-based Development (PBD) KA, but rather PDB "is concerned with the design and development of software applications that reside on specific software platforms. In contrast to general purpose programming, platform-based development takes into account platform-specific constraints. For instance, web programming, multimedia development, mobile computing, app development, and robotics are examples of relevant platforms that provide specific services/APIs/hardware that constrain development." (*cs2013.org, p. 144*)

Table 1: Knowledge Areas of the Associate-Degree CS Transfer Curriculum

Algorithms and Complexity (AL)	Architecture and Organization (AR)
Computational Science (CN)	Discrete Structures (DS)
Graphics and Visualization (GV)	Human-Computer Interaction (HCI)
Information Assurance and Security (IAS)	Information Management (IM)
Networking and Communications (NC)	Operating Systems (OS)
Parallel and Distributed Computing (PD)	Programming Languages (PL)
Software Development Fundamentals (SDF)	Software Engineering (SE)
System Fundamentals (SF)	Social Issues and Professional Practice (SP)

The 16 Knowledge Areas in Table 1 along with the associated Knowledge Units in Table 2 and student learning outcomes comprise the ACM Body of Knowledge for associate-degree transfer curriculum in computer science. The learning outcomes are expressed using action verbs from Bloom's Revised Taxonomy. The Bloom's level is indicated in *[italics]* after each learning outcome. A three-tiered assessment rubric labeled as "emerging", "developed", and "highly developed" provides further clarity to each learning outcome. Typically as the level of student achievement progresses from "emerging" to "highly developed", the level of Bloom's action verb also increases from the lower order thinking skills (LOTS) to the higher order thinking skills (HOTS) in the cognitive domain. The "developed", middle tier indicates the student has met the intended expectation of the learning outcome. For easy referencing throughout the BoK, the cybersecurity-related learning outcomes are highlighted in **bold red font**. See Appendix A for a

consolidated list of all the cybersecurity-related learning outcomes organized by knowledge area and unit.

Table 2: Knowledge Areas and Knowledge Units

Algorithms and Complexity	**Architecture and Organization**
Basic Analysis	Digital Logic and Digital Systems
Algorithmic Strategies	Machine Level Representation of Data
Fundamental Data structures and Algorithms	Assembly Level Machine Organization
Basic Automata, Computability, and Complexity	Memory System Organization and Architecture
Computational Science	**Discrete Structures**
Introduction to Modeling and Simulation	Sets, Relations, and Functions
	Basic Logic
	Proof Techniques
	Basics of Counting
	Graphs and Trees
	Discrete Probability
Graphics and Visualization	**Human Computer Interaction**
Fundamental Concepts	Foundations
	Designing Interaction
Information Assurance and Security	**Information Management**
Foundational Concepts in Security	Information Management Concepts
Principles of Secure Design	Database Systems
Defensive Programming	Data Modeling

Threats and Attacks	
Cryptography	
Web Security	
Secure Software Engineering	
Networking and Communication	**Operating Systems**
Introduction	Overview of Operating Systems
Networked Applications	Operating System Principles
	Concurrency
	Memory Management
	Security and Protection
	Virtual Machines
	Device Management
Parallel and Distributed Computing	**Programming Languages**
Parallelism Fundamentals	Object-Oriented Programming
Communication and Coordination	Functional Programming
	Event-Driven and Reactive Programming
	Basic Type Systems
Software Development Fundamentals	**Software Engineering**
Algorithms and Design	Software Processes
Fundamental Programming Concepts	Software Project Management
Fundamental Data Structures	Tools and Environments
Development Methods	Requirements Engineering

	Software Design
	Software Construction
	Software Verification and Validation
Systems Fundamentals	**Social Issues and Professional Practice**
Computational Paradigms	Social Context
Cross-Layer Communications	Analytical Tools
Parallelism	Professional Ethics
	Intellectual Property
	Privacy and Civil Liberties
	Professional Communication
	Sustainability
	Security Policies, Laws and Computer Crime

Algorithms and Complexity (AL)

This knowledge area defines the central concepts and skills required to design, implement, and analyze algorithms for solving problems. Algorithms are fundamental to computer science and software engineering. The real-world performance of any software system depends on: (1) the algorithms chosen and (2) the suitability and efficiency of the various layers of implementation. Good algorithm design is therefore crucial for the performance of all software systems. Moreover, the study of algorithms provides insight into the intrinsic nature of the problem as well as possible solution techniques independent of programming language, programming paradigm, computer hardware, or any other implementation aspect, such as security. An important part of computing is the ability to select algorithms appropriate to particular purposes and to apply them, recognizing the possibility that no suitable algorithm may exist. This facility relies on understanding the range of algorithms that address an important set of well-defined problems, recognizing their strengths and weaknesses, and their suitability in particular contexts. Efficiency is a pervasive theme throughout this knowledge area. (*adapted from cs2013, p. 55*).

Learning Outcome	**Assessment Rubric**		
AL. Algorithms and Complexity KA	**Emerging**	**Developed**	**Highly Developed**
AL/Basic Analysis KU			
AL-01. Explain the differences among best, average, and worst case behaviors of an algorithm. [*Understanding*]	Label best, average, and worst bounds in a plot of a common function. [*Remembering*]	Explain the differences among best, average, and worst case behaviors of an algorithm. [*Understanding*]	Illustrate the differences among best, average, and worst case behaviors of an algorithm. [*Applying*]
AL-02. Classify the time and space complexity of simple algorithms using Big-O notation. [*Understanding*]	Name some complexity categories of algorithms. [*Remembering*]	Classify the time and space complexity of simple algorithms using Big-O notation. [*Understanding*]	Calculate the time and space complexity for a given algorithm. [*Applying*]
AL-03. Contrast standard complexity classes. [*Analyzing*]	Illustrate a few of the standard complexity classes. [*Applying*]	Contrast standard complexity classes, such as logarithmic, linear, quadratic, and exponential. [*Analyzing*]	Evaluate standard complexity classes to classify algorithms into "efficient" and "inefficient". [*Evaluating*]
AL-04. Execute algorithms on input of various sizes and compare performance. [*Applying*]	Discuss algorithms on input of various sizes and compare performance. [*Understanding*]	Execute algorithms on input of various sizes and compare performance. [*Applying*]	Compare algorithms on input of various sizes and compare performance. [*Analyzing*]
AL/Algorithmic Strategies KU			
AL-05. Implement a recursive solution to a problem. *[Applying]*	Identify a recursive pattern. [*Remembering*]	Implement a recursive solution to a problem. [*Applying*]	Examine a recursive solution to a problem. [*Analyzing*]

AL-06. Apply an appropriate algorithmic approach to a problem. *[Applying]*	Identify an appropriate algorithmic approach to a problem. [*Remembering*]	Apply an appropriate algorithmic approach to a problem, such as greedy, divide-and-conquer, and dynamic programming. *[Applying]*	Analyze the tradeoffs of various algorithmic approaches to a problem. *[Analyzing]*
AL-07. Explain the role of using random/pseudo random number generation in cryptography. [*Understanding*]	Recognize the role of random numbers in computer security. [*Remembering*]	Explain the role of using random/pseudo random number generation in computer security, such as password generation and data encryption? [*Understanding*]	Implement an algorithm that uses random numbers in computer security. [*Applying*]
AL/Fundamental Data Structures and Algorithms KU			
AL-08. Implement basic numerical algorithms. *[Applying]*	Explain basic numerical algorithms. [*Understand*]	Implement basic numerical algorithms, such as finding min, max, and mode. [*Applying*]	Implement complex numerical algorithms. [*Applying*]
AL-09. Implement simple search algorithms. *[Applying]*	Name simple search algorithms. [*Remembering*]	Implement simple search algorithms, such as linear and binary search. [*Applying*]	Compare simple search algorithms. [*Analyzing*]
AL-10. Implement common iterative and recursive sorting algorithms. *[Applying]*	Implement common sorting algorithms. [*Applying*]	Implement common iterative and recursive sorting algorithms, such as bubble sort, mergesort, and quicksort. [*Applying*]	Compare common iterative and recursive sorting algorithms. [*Analyzing*]
AL-11. Implement hash tables, including collision avoidance and resolution. [*Applying*]	Explain the general idea of a hash table. [*Understand*]	Implement hash tables, including collision avoidance and resolution. [*Applying*]	Compare common collision resolution techniques for hash tables. [*Analyzing*]
AL-12. Discuss the runtime and memory efficiency of principal algorithms for sorting, searching, and hashing. *[Understanding]*	Recall the runtime and memory efficiency of principal algorithms. [*Remembering*]	Discuss the runtime and memory efficiency of principal algorithms for sorting, searching, and hashing. [*Understanding*]	Compare the runtime and memory efficiency of principal algorithms for sorting, searching, and hashing. [*Analyzing*]
AL-13. Investigate factors other than computational efficiency that influence the choice of	List other factors other than computational efficiency that ought to	Investigate factors other than computational	Critique factors other than computational efficiency that

algorithms. *[Applying]*	be considered when choosing an algorithm. [*Remembering*]	efficiency that influence the choice of algorithms, such as programming time, known vulnerabilities, and maintainability. [*Applying*]	influence the choice of algorithms. [*Evaluating*]
AL-14. Solve problems using fundamental graph algorithms, including depth-first and breadth-first search. *[Applying]*	State the differences between depth-first and breadth-first search. [*Remembering*]	Solve problems using fundamental graph algorithms, including depth-first and breadth-first search. [*Applying*]	Design algorithms to find strongly connected components based on the principles of depth-first search. [*Creating*]
AL-15. Contrast various data structures to solve a given problem. *[Analyzing]*	Explain which data structure (array, list, set, map, stack, queue, hash table, tree, or graph) is appropriate to solve a given problem. [*Understanding*]	Contrast various data structures, such as array, list, set, map, stack, queue, hash table, tree, or graph, to solve a given problem. [*Analyzing*]	Justify a choice of data structure to solve a given problem. [*Evaluating*]
AL-16. Summarize the security vulnerabilities in various data structures. *[Understanding]*	List some of the security vulnerabilities in various data structures. [*Remembering*]	Summarize the security vulnerabilities in various data structures, such as out-of-bounds and buffer overflow errors. [*Understanding*]	Investigate the security vulnerabilities in various data structures. [*Applying*]
AL/Basic Automata, Computability, and Complexity KU			
AL-17. Write a regular expression to match a pattern. *[Applying]*	Explain the use of regular expressions in pattern matching. *[Understanding]*	Write a regular expression to match a pattern. *[Applying]*	Write a regular expression to perform complex pattern matching. *[Applying]*
AL-18. Discuss finite state machines. *[Understanding]*	Recognize a finite state machine. *[Remembering]*	Discuss the concept of finite state machines. [*Understanding*]	Diagram a finite state machine. *[Applying]*
AL-19. Explain why the halting problem has no algorithmic solution. *[Understanding]*	Recognize that some problems have no algorithmic solution. *[Remembering]*	Explain why the halting problem has no algorithmic solution. *[Understanding]*	Illustrate a proof of the halting problem. *[Applying]*

Architecture and Organization (AR)

Computing professionals should not regard the computer as just a black box that executes programs by magic. The knowledge area Architecture and Organization builds on Systems Fundamentals (SF) to develop a deeper understanding of the hardware environment upon which all computing is based, and the interface it provides to higher software layers. Students should acquire an understanding and appreciation of a computer system's functional components, their characteristics, performance, and interactions, and, in particular, the challenge of harnessing parallelism to sustain performance improvements now and into the future. Students need to understand computer architecture to develop programs that can achieve high performance through a programmer's awareness of parallelism and latency. In selecting a system to use, students should be able to understand the tradeoff among various components, such as CPU clock speed, cycles per instruction, memory size, and average memory access time. (*adapted from cs2013, p. 62*)

Learning Outcome	**Assessment Rubric**		
AR. Architecture and Organization KA	**Emerging**	**Developed**	**Highly Developed**
AR/Digital Logic and Digital Systems KU			
AR-01. Diagram the digital components of computing architecture. [*Applying*]	Discuss some of the digital components of computing architecture. [*Understanding*]	Diagram the digital components of computing architecture, such as digital gates, registers, and memory. [*Applying*]	Evaluate the digital component design of a computing architecture for accuracy. [*Evaluating*]
AR/Machine Level Representation of Data KU			
AR-02. Explain reasons for using alternative formats to represent numerical data. [*Understanding*]	Recognize the reasons for using alternative formats to represent numerical data. *[Remembering]*	Explain reasons for using alternative formats to represent numerical data. [*Understanding*]	Apply alternative formats to represent numerical data. [*Applying*]
AR-03. Explain how fixed-length number representations could affect accuracy, precision, and vulnerabilities. *[Understanding]*	Define fixed-length number representations. *[Remembering]*	Explain how fixed-length number representations could affect accuracy, precision, and vulnerabilities. [*Understanding*]	Illustrate how fixed-length number representations could affect accuracy, precision, and vulnerabilities. [*Applying*]
AR-04. Describe the internal representation of non-numeric data.	Define the internal representation of non-numeric data	Describe the internal representation of non-numeric data, such as	Illustrate internal representation of non-numeric data.

[Understanding]	*[Remembering]*	characters, strings, records, and arrays. *[Understanding]*	[*Applying*]
AR-05. Convert numerical data from one format to another. *[Understanding]*	Describe how to convert numerical data from one format to another. *[Remembering]*	Convert numerical data from one format to another, such as, negative integers into sign-magnitude and two's-complement representations. *[Understanding]*	Compare and contrast different methods for converting numerical data from one format to another. [*Applying*]
AR/Assembly Level Machine Organization KU			
AR-06. Explain the organization of the classical von Neumann machine and its major functional units. *[Understanding]*	Define the organization of the classical von Neumann machine and its major functional units. *[Remembering]*	Explain the organization of the classical von Neumann machine and its major functional units. *[Understanding]*	Diagram the organization of the classical von Neumann machine and its major functional units. *[Applying]*
AR-07. Demonstrate how high-level language patterns map to assembly/machine language, including subroutine calls. *[Understanding]*	Recognize how high-level language patterns map to assembly/machine language, including subroutine calls. *[Remembering]*	Demonstrate how high-level language patterns map to assembly/machine language, including subroutine calls. *[Understanding]*	Diagram high-level language patterns map to assembly/machine language, including subroutine calls. *[Applying]*
AR-08. Write simple assembly language program segments. *[Applying]*	Explain simple assembly language program segments. *[Understanding]*	Write simple assembly language program segments. *[Applying]*	Write more complex assembly language program segments. *[Applying]*
AR-09. Explain the basic concepts of interrupts and I/O operations. *[Understanding]*	List basic concepts of interrupts and I/O operations. *[Remembering]*	Explain the basic concepts of interrupts and I/O operations. *[Understanding]*	Implement basic concepts of interrupts and I/O operations. *[Applying]*
AR/Memory System Organization and Architecture KU			
AR-10. Identify the main types of memory technology. *[Remembering]*	Name some types of memory technology. *[Remembering]*	Identify the main types of memory technology, such as SRAM, DRAM, Flash, magnetic disk. *[Remembering]*	Differentiate the main types of memory technology and discuss their relative cost and performance. *[Understanding]*
AR-011. Explain the effect of memory latency on execution time across the memory hierarchy. *[Understanding]*	Recognize the effect of memory latency on running time. *[Remembering]*	Explain the effect of memory latency on execution time across the memory hierarchy. *[Understanding]*	Calculate the effect of memory latency on execution time across the memory hierarchy. *[Applying]*

Computational Science (CN)

Computational science is a field of applied computer science, that is, the application of computer science to solve problems across a range of disciplines, such as molecular and fluid dynamics, celestial mechanics, economics, biology, geology, medicine, and social network analysis. Fundamental concepts of computational science are germane to every computer scientist, such as modeling and simulation. This area offers exposure to many valuable ideas and techniques, including precision of numerical representation, error analysis, numerical techniques, parallel architectures and algorithms, quantum computing, modeling and simulation, information visualization, software engineering, and optimization. Topics relevant to computational science include fundamental concepts in program construction (SDF/Fundamental Programming Concepts), algorithm design (SDF/Algorithms and Design), program testing (SDF/Development Methods), data representations (AR/Machine Representation of Data), and basic computer architecture (AR/Memory System Organization and Architecture). (*adapted from cs2013, p. 70*)

Learning Outcome	**Assessment Rubric**		
CN. Computational Science KA	**Emerging**	**Developed**	**Highly Developed**
CN/Introduction to Modeling and Simulation KU			
CN-01. Illustrate the concepts of modeling and abstraction with respect to problem solving. *[Applying]*	Define the concepts of modeling and abstraction with respect to problem solving. *[Remembering]*	Illustrate the concepts of modeling and abstraction with respect to problem solving. *[Applying]*	Contrast the concepts of modeling and abstraction with respect to problem solving. *[Analyzing]*
CN-02. Describe the relationship between modeling and simulation. *[Understanding]*	List the relationships between modeling and simulation. *[Remembering]*	Describe the relationship between modeling and simulation. *[Understanding]*	Illustrate the relationship between modeling and simulation. *[Applying]*
CN-03. Differentiate among the different types of simulations, including physical simulations, human-guided simulations, and virtual reality. *[Understanding]*	Identify the different types of simulations, including physical simulations, human-guided simulations, and virtual reality. *[Remembering]*	Differentiate among the different types of simulations, including physical simulations, human-guided simulations, and virtual reality. *[Understanding]*	Investigate the different types of simulations, including physical simulations, human-guided simulations, and virtual reality. *[Applying]*
CN-04. Explain and give examples of the benefits of simulation and modeling in a range of important application areas. *[Understanding]*	Identify the benefits of simulation and modeling in a range of important application areas. *[Remembering]*	Explain the benefits of simulation and modeling in a range of important application areas. *[Understanding]*	Investigate and show, using examples, the benefits of simulation and modeling in a range of important application areas. *[Applying]*

Discrete Structures (DS)

Discrete structures serve as a foundation for many areas in computer science. Discrete structures include the study of logic, set theory, graph theory, and probability theory. The concepts studied in discrete structures are pervasive throughout the areas of data structures and algorithms, but also appear elsewhere in computer science. For example, an ability to create and understand a proof—either a formal symbolic proof or a less formal but still mathematically rigorous argument—is important in virtually every area of computer science, including formal specification, verification, databases, and cryptography. Graph theory concepts are used in networks, operating systems, and compilers. Set theory concepts are used in software engineering and in databases. Probability theory is used in intelligent systems, networking, and a number of computing applications. *(adapted from cs2013, p. 76)*

Learning Outcome	**Assessment Rubric**		
DS. Discrete Structures KA	**Emerging**	**Developed**	**Highly Developed**
DS/Sets, Relations, and Functions KU			
DS-01. Explain with examples the basic terminology of functions, relations, and sets. *[Understanding]*	Identify the defining features of functions, relations, and sets. *[Remembering]*	Explain with examples the basic terminology of functions, relations, and sets. *[Understanding]*	Use function, relations, and set terminology in a correct and meaningful way. *[Applying]*
DS-02. Perform the operations associated with sets, functions, and relations. *[Applying]*	Describe the operations associated with sets, functions, and relations. *[Understanding]*	Perform the operations associated with sets, functions, and relations. *[Applying]*	Compare the operations of sets, functions, and relations. *[Analyzing]*
DS-03. Compare practical examples to the appropriate set, function, or relation model, and interpret the associated operations and terminology in context. *[Analyzing]*	Implement a solution to a programming problem using a particular set, function, or relation model. *[Applying]*	Compare practical examples to the appropriate set, function, or relation model, and interpret the associated operations and terminology in context. *[Analyzing]*	Justify the choice of a particular set, function, or relation model. *[Evaluating]*
DS/Basic Logic KU			
DS-04. Convert logical statements from informal language to propositional and predicate logic expressions. *[Understanding]*	Recognize the relationship between logical statements from informal language and propositional and predicate logic expressions. *[Remembering]*	Convert logical statements from informal language to propositional and predicate logic expressions. *[Understanding]*	Produce propositional and predicate logic expressions from a given logical statement from an informal language. *[Applying]*
DS-05. Apply formal logic proofs	Describe the steps in	Apply formal logic	Compare different

and/or informal, but rigorous, logical reasoning to real problems such as predicting the behavior of software or solving problems such as puzzles. *[Applying]*	formal logic proofs and/or informal logical reasoning to solve real problems. *[Understanding]*	proofs and/or informal, but rigorous, logical reasoning to real problems such as predicting the behavior of software or solving problems such as puzzles. *[Applying]*	logic proofs and informal logical reasoning to determine correct methods to solve real problems. *[Analyzing]*
DS-06. Use the rules of inference to construct proofs in propositional and predicate logic. *[Applying]*	Discuss the rules of inference to construct proofs in propositional and predicate logic. *[Understanding]*	Use the rules of inference to construct proofs in propositional and predicate logic. *[Applying]*	Analyze the rules of infe construct proofs in prop predicate logic. *[Analyzing]*
DS-07. Describe how symbolic logic can be used to model real-life situations or applications, including those arising in computing contexts such as software analysis (e.g. program correctness), database queries, and algorithms. *[Understanding]*	List ways that symbolic logic can be used to model real-life situations or applications. *[Remembering]*	Describe how symbolic logic can be used to model real-life situations or applications, including those arising in computing contexts such as software analysis (e.g. program correctness), database queries, and algorithms. *[Understanding]*	Use symbolic logic to model real-life situations. *[Applying]*
DS-08. Apply formal methods of symbolic propositional and predicate logic, such as calculating validity of formulae and computing normal forms. *[Applying]*	Demonstrate formal methods of symbolic propositional and predicate logic. *[Understanding]*	Apply formal methods of symbolic propositional and predicate logic, such as calculating validity of formulae and computing normal forms. *[Applying]*	Distinguish between formal methods of propositional and predicate logic to determine the most effective solutions to a given problem. *[Analyzing]*
DS-09. Describe the strengths and limitations of propositional and predicate logic. *[Understanding]*	List the strengths and limitations of propositional and predicate logic. *[Remembering]*	Describe the strengths and limitations of propositional and predicate logic. *[Understanding]*	Illustrate the strengths and limitations of propositional and predicate logic. *[Applying]*
DS/Proof Techniques KU			
DS-10. Outline the basic structure of each proof technique, including direct proof, proof by contradiction, and induction. *[Analyzing]*	Use the basic structure of each proof technique to solve a problem. *[Applying]*	Outline the basic structure of each proof technique, including direct proof, proof by contradiction, and induction. *[Analyzing]*	Choose the most effective proof technique to solve a problem. *[Evaluating]*

DS-11. Apply each of the proof techniques (direct proof, proof by contradiction, and induction) correctly in the construction of a sound argument. *[Applying]*	Demonstrate each of the proof techniques by correctly constructing a sound argument. *[Understanding]*	Apply each of the proof techniques (direct proof, proof by contradiction, and induction) correctly in the construction of a sound argument. *[Applying]*	Use each of the proof techniques correctly in the construction of a sound argument. *[Applying]*
DS-12. Deduce the best type of proof for a given problem. *[Analyzing]*	Compare the different proof methods. *[Analyzing]*	Deduce the best type of proof for a given problem. *[Analyzing]*	Construct a correct proof using the best method for a given problem. *[Creating]*
DS-13. Explain the parallels between ideas of mathematical and/or structural induction to recursion and recursively defined structures. *[Understanding]*	Identify the parallels between ideas of mathematical and/or structural induction to recursion and recursively defined structures. *[Remembering]*	Explain the parallels between ideas of mathematical and/or structural induction to recursion and recursively defined structures. *[Understanding]*	Illustrate the parallels between ideas of mathematical and/or structural induction to recursion and recursively defined structures. *[Applying]*
DS-14. Explain the relationship between weak and strong induction and give examples of the appropriate use of each. *[Understanding]*	Identify the relationship between weak and strong induction. *[Remembering]*	Explain the relationship between weak and strong induction and give examples of the appropriate use of each. *[Understanding]*	Solve problems using both weak and strong induction. *[Applying]*
DS/Basics of Counting KU			
DS-15. Apply counting arguments, including sum and product rules, inclusion-exclusion principle and arithmetic/geometric progressions. *[Applying]*	Describe counting arguments. *[Understanding]*	Apply counting arguments, including sum and product rules, inclusion-exclusion principle and arithmetic/geometric progressions. *[Applying]*	Outline counting arguments. *[Analyzing]*
DS-16. Apply the pigeonhole principle in the context of a formal proof. *[Applying]*	Demonstrate the pigeonhole principle. *[Understanding]*	Apply the pigeonhole principle in the context of a formal proof. *[Applying]*	Analyze the pigeonhole principle in the context of a formal proof. *[Analyzing]*
DS-17. Calculate permutations and combinations of a set, and interpret the meaning in the context of the particular application.	Explain the calculation of permutations and combinations of a set. *[Understanding]*	Calculate permutations and combinations of a set, and interpret the meaning in the	Discriminate between computation of st permutations and combinations. *[Analyzing]*

[Applying]		context of the particular application. *[Applying]*	
DS-18. Compare real-world applications to appropriate counting formalisms. *[Analyzing]*	Use counting formalisms to solve real-world applications. *[Applying]*	Compare real-world applications to appropriate counting formalisms, such as determining the number of ways to arrange people around a table, subject to constraints on the seating arrangement, or the number of ways to determine certain hands in cards (e.g., a full house). *[Analyzing]*	Choose appropriate counting formalisms to solve real-world applications. *[Evaluating]*
DS-19. Solve a variety of basic recurrence relations. *[Applying]*	Demonstrate a variety of basic recurrence relations. *[Understanding]*	Solve a variety of basic recurrence relations. *[Applying]*	Compare a variety of basic recurrence relations. *[Analyzing]*
DS-20. Analyze a problem to determine underlying recurrence relations. *[Analyzing]*	Carry out a problem with an underlying recurrence relation. *[Applying]*	Analyze a problem to determine underlying recurrence relations. *[Analyzing]*	Evaluate a problem with an underlying recurrence relation. *[Evaluating]*
DS-21. Perform computations involving modular arithmetic. *[Applying]*	Discuss computations involving modular arithmetic. *[Understanding]*	Perform computations involving modular arithmetic. *[Applying]*	Examine computations involving modular arithmetic. *[Analyzing]*
DS/Graphs and Trees KU			
DS-22. Illustrate the basic terminology of graph theory including properties and special cases for each type of graph/tree. *[Applying]*	Describe the basic terminology of graph theory. *[Understanding]*	Illustrate the basic terminology of graph theory including properties and special cases for each type of graph/tree. *[Applying]*	Outline the basic terminology of graph theory. *[Analyzing]*
DS-23. Demonstrate different traversal methods for trees and graphs, including pre-, post-, and in-order traversal of trees. *[Understanding]*	List the different traversal methods for trees and graphs *[Remembering]*	Demonstrate different traversal methods for trees and graphs, including pre-, post-, and in-order traversal of trees. *[Understanding]*	Execute different traversal methods for trees and graphs. *[Applying]*

DS-24. Solve a variety of real-world problems in computer science using appropriate forms of graphs and trees, such as representing a network topology or the organization of a hierarchical file system. *[Applying]*	Discuss a variety of real-world problems in computer science using appropriate forms of graphs and trees. *[Understanding]*	Solve a variety of real-world problems in computer science using appropriate forms of graphs and trees, such as representing a network topology or the organization of a hierarchical file system. *[Applying]*	Distinguish between real-world problems solvable by using graphs and trees. *[Analyzing]*
DS-25. Implement and use balanced trees and B-trees. *[Applying]*	Explain balanced trees and B-trees. *[Understanding]*	Implement and use balanced trees and B-trees. *[Applying]*	Analyze the use of balanced trees and B-trees. *[Analyzing]*
DS-26. Implement graph algorithms. *[Applying]*	Classify graph algorithms. *[Understanding]*	Implement graph algorithms, such as graph search, union-find, minimum spanning trees, and shortest paths. *[Applying]*	Categorize different implementations of graph algorithms. *[Analyzing]*
DS-27. Demonstrate how concepts from graphs and trees appear in data structures, algorithms, proof techniques (structural induction), and counting. *[Understanding]*	Identify how concepts from graphs and trees appear in data structures, algorithms, proof techniques, and counting. *[Remembering]*	Demonstrate how concepts from graphs and trees appear in data structures, algorithms, proof techniques (structural induction), and counting. *[Understanding]*	Implement data structures, algorithms, proof techniques, and counting using graphs and trees. *[Applying]*
DS-28. Describe binary search trees and AVL trees. *[Understanding]*	Define binary search and AVL trees. *[Remembering]*	Describe binary search trees and AVL trees. [Understanding]	Apply binary search and AVL trees. *[Applying]*
DS-29. Explain complexity in the ideal and in the worst case scenario for both implementations. *[Understanding]*	State complexity in the ideal and in the worst case scenario for both implementations. *[Remembering]*	Explain complexity in the ideal and in the worst case scenario for both implementations. *[Understanding]*	Calculate complexity in the ideal and in the worst case scenario for both implementations. *[Applying]*
DS/Discrete Probability KU			
DS-30. Calculate probabilities of events and expectations of random variables for elementary problems. *[Applying]*	Exemplify probabilities of events and expectations of random variables for elementary problems	Calculate probabilities of events and expectations of random variables for elementary problems	Examine probabilities of events and expectations of random variables for elementary problems

	such as games of chance. *[Understanding]*	such as games of chance. *[Applying]*	such as games of chance. *[Analyzing]*
DS-31. Differentiate between dependent and independent events. *[Understanding]*	Identify dependent and independent events. *[Remembering]*	Differentiate between dependent and independent events. *[Understanding]*	Illustrate dependent and independent events. *[Applying]*
DS-32. Explain the significance of binomial distribution in probabilities. *[Understanding]*	Recognize the notation and parameters that a binomial distribution has. *[Remembering]*	Explain the significance of binomial distribution in probabilities. *[Understanding]*	Calculate the probabilities from a binomial distribution. *[Applying]*
DS-33. Apply Bayes Theorem to determine conditional probabilities in a problem. *[Applying]*	Explain Bayes Theorem to determine conditional probabilities in a problem. *[Understanding]*	Apply Bayes Theorem to determine conditional probabilities in a problem. *[Applying]*	Outline Bayes Theorem to determine conditional probabilities in a problem. *[Analyzing]*
DS-34. Apply the tools of probability to solve problems such as the average case analysis of algorithms or analyzing hashing. *[Applying]*	Discuss the tools of probability to solve problems such as the average case analysis of algorithms or analyzing hashing. *[Understanding]*	Apply the tools of probability to solve problems such as the average case analysis of algorithms or analyzing hashing. *[Applying]*	Analyze the tools of probability in solving problems such as the average case analysis of algorithms or analyzing hashing. *[Analyzing]*

Graphics and Visualization (GV)

Computer graphics is the term commonly used to describe the computer generation and manipulation of images. It is the science of enabling visual communication through computation, including cartoons, film special effects, videogames, medical imaging, engineering, as well as scientific, information, and knowledge visualization. Traditionally, graphics at the undergraduate level has focused on rendering, linear algebra, and phenomenological approaches. More recently, the focus has begun to include physics, numerical integration, scalability, and special-purpose hardware. In order for students to become adept at the use and generation of computer graphics, many implementation-specific issues must be addressed, such as file formats, hardware interfaces, and application program interfaces. These issues change rapidly, and the description that follows attempts to avoid being overly prescriptive about them. The study of Graphics and Visualization is divided into several interrelated fields:

- **Fundamentals**: Computer graphics depends on an understanding of how humans use vision to perceive information and how information can be rendered on a display device. Every computer scientist should have some understanding of where and how graphics can be appropriately applied as well as the fundamental processes involved in display rendering.
- **Modeling**: Information to be displayed must be encoded in computer memory in some form, often in the form of a mathematical specification of shape and form.
- **Rendering**: Rendering is the process of displaying the information contained in a model.
- Animation: Animation is the rendering in a manner that makes images appear to move and the synthesis or acquisition of the time variations of models.
- **Visualization**: The field of visualization seeks to determine and present underlying correlated structures and relationships in data sets from a wide variety of application areas. The prime objective of the presentation should be to communicate the information in a dataset so as to enhance understanding.
- **Computational Geometry**: Computational Geometry is the study of algorithms that are stated in terms of geometry.

(*adapted from cs2013, p. 82*)

Learning Outcome	**Assessment Rubric**		
GV. Graphics and Visualization KA	**Emerging**	**Developed**	**Highly Developed**
GV/Fundamental Concepts KU			
GV-01. Explain the progression of Dimension and Coordinate Systems. *[Understanding]*	Define dimensional and coordinate Systems. *[Remembering]*	Explain how to use dimensions and coordinate systems. *[Understanding]*	Compare transformation and changes in dimension and coordinate systems for 2D and 3D design. *[Analyzing]*

GV-02. Describe common uses of digital presentation to human senses. *[Understanding]*	List a variety of digital presentation media in relationship to human senses. *[Remembering]*	Describe common uses of digital presentation to human senses such as computer graphics, sound, haptic devices. *[Understanding]*	Choose appropriate digital presentation options based on project need. *[Evaluating]*
GV-03. Describe color models and their use in computer graphics. *[Understanding]*	Identify color models. *[Remembering]*	Describe color models and their use in computer graphics. *[Understanding]*	Contrast color models and their use in computer graphics. *[Analyzing]*
GV-04. Differentiate multimedia file types, resolution needs, conversion, and appropriate use. *[Understanding]*	List the advantages and disadvantages to pixel vs. vector image structures. *[Remembering]*	Differentiate multimedia file types, resolution needs, conversion, and appropriate use. *[Understanding]*	Analyze image types according to output choices. *[Analyzing]*
GV-05. Illustrate the principle of information hiding through steganography in images, messages, videos, or other media files. *[Applying]*	Explain the principle of information hiding through steganography. *[Understanding]*	Illustrate the principle of information hiding through steganography in images, messages, videos or other media files. *[Applying]*	Apply a given steganography algorithm to conceal a media file, such as an image, message, video, or other media file. *[Applying]*

Human-Computer Interaction (HCI)

Human-computer interaction (HCI) studies cognitive science and human factors engineering as applied to computer science. HCI is concerned with designing interactions between human activities and the computational systems that support them, and with constructing interfaces to afford those interactions. Interaction between users and computational artifacts occurs at an interface that includes both software and hardware. Thus interface design impacts the software lifecycle in that it should occur early; the design and implementation of core functionality can influence the user interface – for better or worse. Because it deals with people as well as computational systems, the HCI knowledge area considers cultural, social, organizational, cognitive and perceptual issues. Consequently it draws on a variety of disciplinary traditions, including psychology, ergonomics, graphic and product design, anthropology and engineering. (*adapted from cs2013, p. 89*)

Learning Outcome	**Assessment Rubric**		
HCI. Human Computer Interaction KA	**Emerging**	**Developed**	**Highly Developed**
HCI/Foundations KU			
HCI-01. Discuss the importance of human-centered software development. *[Understanding]*	Define human-centered software. *[Remembering]*	Discuss the importance of human-centered software development.. *[Understanding]*	Examine the importance of Human-centered software. *[Analyzing]*
HCI-02. Use a conceptual vocabulary for analyzing human interaction with software applications. *[Applying]*	Memorize vocabulary terms specific to human- computer interaction. *[Remembering]*	Use a conceptual vocabulary for analyzing human interaction with software applications, such as affordance, conceptual model, and feedback. *[Applying]*	Categorize a conceptual vocabulary for analyzing human interaction with software applications. *[Analyzing]*
HCI-03. Implement a simple usability test for an existing software application. *[Applying]*	Summarize usability testing. *[Understanding]*	Implement a simple usability test for an existing software application. *[Applying]*	Design a usability test for an existing software application *[Creating]*
HCI-04. Investigate the issues of trust in HCI, including examples of both high and low trust systems. *[Applying]*	List design elements that make a human-computer interface trustworthy. *[Remembering]*	Investigate the issues of trust in HCI, including examples of both high and low trust systems. *[Applying]*	Critique interface designs between high trust and low trust. *[Evaluating]*

HCI/Designing Interaction KU			
HCI-05. Write a simple application that uses a modern graphical user interface. *[Applying]*	Summarize the components of a modern graphical user interface. *[Understanding]*	Write a simple application that uses a modern graphical user interface. *[Applying]*	Examine a complete application that uses a modern graphical interface and includes technical documentation. *[Analyzing]*
HCI-06. Use at least one national or international user interface design standard in a simple application. *[Applying]*	Identify national and international user interface design standards. *[Remembering]*	Use at least one national or international user interface design standard in a simple application. *[Applying]*	Compare among national and international user interface design standards. *[Analyzing]*
HCI-07. Analyze the interface needs of an identified user group. *[Analyzing]*	Describe common user interface needs. *[Understanding]*	Analyze the interface needs of an identified user group. *[Analyzing]*	Assess the appropriateness of design standards in meeting specified user interface needs. *[Evaluating]*
HCI-08. Illustrate the interaction between a security mechanism and its usability. *[Applying]*	Discuss potential usability issues related to a security mechanism. *[Understanding]*	Illustrate the interaction between a security mechanism and its usability. *[Applying]*	Design a user interface that minimizes the tradeoffs between usability and security. *[Creating]*

Information Assurance and Security (IAS)

In CS2013, the Information Assurance and Security KA was added to the Body of Knowledge in recognition of the world's reliance on information technology and its critical role in computer science education. Information assurance and security as a domain is the set of controls and processes both technical and policy intended to protect and defend information and information systems by ensuring their confidentiality, integrity, and availability, and by providing for authentication and non-repudiation. The concept of assurance also carries an attestation that current and past processes and data are valid. Both assurance and security concepts are needed to ensure a complete perspective. Information assurance and security education, includes all efforts to prepare students with the needed knowledge, skills, and abilities to protect our information systems and attest to the assurance of the past and current state of processes and data. The importance of security concepts and topics has emerged as a core requirement in the Computer Science discipline, much like the importance of performance concepts has been for many years. The Information Assurance and Security KA is unique among the set of KAs presented here given the manner in which the topics are pervasive throughout other Knowledge Areas. (*adapted from cs2013, p. 97*)

Learning Outcome	Assessment Rubric		
IAS. Information Assurance and Security KA	**Emerging**	**Developed**	**Highly Developed**
IAS/Foundational Concepts in Security KU			
IAS-01. Recognize the importance of security as a continuous process and its balancing nature between protection mechanisms and availability of data and information. *[Remembering]*	Recognize the importance of security. *[Remembering]*	Recognize the importance of security as a continuous process and its balancing nature between protection mechanisms and availability of data and information. *[Remembering]*	Explain the importance of security as a continuous process and its balancing nature between protection mechanisms and availability of data and information. *[Understanding]*
IAS-02. Describe the concepts of risk, threats, vulnerabilities, attack vectors, and exploits (including the fact that there is no such thing as perfect security). *[Understanding]*	Define the concepts of risk, threats, vulnerabilities, attack vectors, and exploits (including the fact that there is no such thing as perfect security). *[Remembering]*	Describe the concepts of risk, threats, vulnerabilities, attack vectors, and exploits (including the fact that there is no such thing as perfect security). *[Understanding]*	Apply the concepts of risk, threats, vulnerabilities, attack vectors, and exploits (including the fact that there is no such thing as perfect security) to a scenario. *[Applying]*
IAS-03. Explain the importance of security controls and countermeasures to minimize security risk and exposure. *[Understanding]*	Recognize the importance of security controls and countermeasures to minimize security risk and exposure. *[Remembering]*	Explain the importance of security controls and countermeasures to minimize security risk and exposure. *[Understanding]*	Implement one or more security controls and countermeasures to minimize security risk and exposure. *[Applying]*
IAS-04. Analyze the tradeoffs of balancing security properties (Confidentiality, Integrity, Availability, as well as Authentication, Authorization, Access, Authenticity, Non-Repudiation, Privacy). *[Analyzing]*	Investigate the tradeoffs of balancing security properties (Confidentiality, Integrity, Availability, as well as Authentication, Authorization, Access, Authenticity, Non-Repudiation, Privacy). *[Applying]*	Distinguish the tradeoffs of balancing security properties (Confidentiality, Integrity, Availability, as well as Authentication, Authorization, Access, Authenticity, Non-Repudiation, Privacy). *[Analyzing]*	Evaluate the tradeoffs of balancing security properties (Confidentiality, Integrity, Availability, as well as Authentication, Authorization, Access, Authenticity, Non-Repudiation, Privacy). *[Evaluating]*
IAS-05. Explain the concepts of trust and trustworthiness. *[Understanding]*	Define the concepts of trust and trustworthiness. *[Remembering]*	Explain the concepts of trust and trustworthiness. *[Remembering]*	Diagram trust relationships. *[Applying]*

IAS-06. Describe important ethical issues to consider in security. *[Understanding]*	Identify important ethical issues to consider in security. *[Remembering]*	Describe important ethical issues to consider in security. *[Understanding]*	Investigate important ethical issues to consider in security. *[Applying]*
IAS-07. Investigate risks to privacy and anonymity in technology. *[Applying]*	Describe risks to privacy and anonymity in technology. *[Understanding]*	Investigate risks to privacy and anonymity in technology. *[Applying]*	Analyze risks to privacy and anonymity in technology. *[Analyzing]*
IAS-08. Apply cybersecurity principles to a changing landscape. *[Applying]*	Summarize cybersecurity principles and how they are affected by a changing landscape. *[Understanding]*	Apply cybersecurity principles to a changing landscape. *[Applying]*	Examine cybersecurity principles in a changing landscape. *[Analyzing]*
IAS-09. Demonstrate the key role risk management frameworks play in identifying, assessing, prioritizing, and controlling risks that an organization's assets face. *[Understanding]*	Recognize the key role risk management frameworks play in identifying, assessing, prioritizing, and controlling risks that an organization's assets face. *[Remembering]*	Demonstrate the key role risk management frameworks play a key role in identifying, assessing, prioritizing, and controlling risks that an organization's assets face. *[Understanding]*	Carry out an example of identifying, assessing, prioritizing, and controlling risk to illustrate how risk management frameworks play a key role in protecting an organization's assets. *[Applying]*
IAS/Principles of Secure Design KU			
IAS-10. Describe the principles of secure design. *[Understanding]*	Define the principles of secure design. *[Remembering]*	Describe the principles of secure design, such as least privilege, isolation, fail-safe, and deny-by-default. *[Understanding]*	Examine the principles of secure design given a scenario. [Analyzing]
IAS-11. Discuss the implications of relying on open design or the secrecy of design for security. *[Understanding]*	Recognize the implications of relying on open design or the secrecy of design for security. *[Remembering]*	Discuss the implications of relying on open design or the secrecy of design for security. *[Understanding]*	Illustrate the implications of relying on open design or the secrecy of design for security. *[Applying]*
IAS-12. Explain the goals of end-to-end data security. *[Understanding]*	List some of the goals of end-to-end data security. *[Remembering]*	Explain the goals of end-to-end data security. *[Understanding]*	Illustrate using examples the goals of end-to-end data security. *[Applying]*
IAS-13. Discuss the benefits of having multiple layers of defenses (Defense In Depth). *[Understanding]*	Recognize one or more of the benefits of having multiple layers of defenses (Defense In Depth). *[Remembering]*	Discuss the benefits of having multiple layers of defenses (Defense In Depth). *[Understanding]*	Implement multiple layers of defenses (Defense In Depth) for a given scenario *[Applying]*

IAS-14. For each stage in the lifecycle of a product, investigate what security considerations should be evaluated. *[Applying]*	For each stage in the lifecycle of a product, describe what security considerations should be evaluated. *[Understanding]*	For each stage in the lifecycle of a product, investigate what security considerations should be evaluated. *[Applying]*	For each stage in the lifecycle of a product, analyze what security considerations should be evaluated. *[Analyzing]*
IAS-15. Recognize the tradeoffs associated with designing security into a product. *[Remembering]*	Recognize some of the tradeoffs associated with designing security into a product. *[Remembering]*	Recognize the tradeoffs associated with designing security into a product. *[Remembering]*	Describe the tradeoffs associated with designing security into a product including cost and benefits. *[Understanding]*
IAS/Defensive Programming KU			
IAS-16. Implement input validation and data sanitization in applications as necessary considering adversarial control of the input channel. *[Understanding]*	Explain the importance of input validation and data sanitization in applications considering adversarial control of the input channel. *[Understanding]*	Implement input validation and data sanitization in applications as necessary considering adversarial control of the input channel. *[Applying]*	Analyze the effectiveness of input validation and data sanitization implemented in applications considering adversarial control of the input channel. *[Analyzing]*
IAS-17. Explain the tradeoffs of developing a program in a type-safe language. *[Understanding]*	List some of the tradeoffs of developing a program in a type-safe language. *[Remembering]*	Explain the tradeoffs of developing a program in a type-safe language. *[Understanding]*	Compare and contrast the tradeoffs of developing a program in a type-safe language. *[Analyzing]*
IAS-18. Implement programs that properly handle exceptions and error conditions. *[Applying]*	Explain the importance of writing programs that properly handle exceptions and error conditions. *[Understanding]*	Implement programs that properly handle exceptions and error conditions. *[Applying]*	Analyze the implementation of exception handling in programs and error conditions. *[Analyzing]*
IAS-19. Recognize the need to update software to fix security vulnerabilities. *[Understanding]*	Recognize the need to update software. *[Understanding]*	Recognize the need to update software to fix security vulnerabilities. *[Understanding]*	Demonstrate the need to update software to fix security vulnerabilities. *[Understanding]*
IAS/Threats and Attacks KU			
IAS-20. Identify likely attack types against standalone and networked software systems. *[Remembering]*	Identify some attack types against software systems. *[Remembering]*	Identify likely attack types against standalone and networked software systems. *[Remembering]*	Discuss likely attack types against standalone and networked software systems. *[Understanding]*

IAS-21. Describe risks to privacy and anonymity in information systems. *[Understanding]*	Name risks to privacy and anonymity in information systems. *[Remembering]*	Describe risks to privacy and anonymity in information systems. *[Understanding]*	Investigate risks to privacy and anonymity in information systems. *[Applying]*
IAS-22. Discuss the key principles, such as membership and trust, of social engineering. *[Understanding]*	Recognize the key principles, such as membership and trust, of social engineering. *[Remembering]*	Discuss the key principles, such as membership and trust, of social engineering. *[Understanding]*	Illustrate the key principles, such as membership and trust, of social engineering. *[Applying]*
IAS/Cryptography KU			
IAS-23. Explain the purpose of cryptography and how it is used to secure data. *[Understanding]*	State the purpose of cryptography. *[Remembering]*	Explain the purpose of cryptography and how it is used to secure data. *[Understanding]*	Implement cryptography to secure data. *[Applying]*
IAS-24. Define key terms in cryptology. *[Remembering]*	List some key terms in cryptology. *[Remembering]*	Define key terms in cryptology, including cryptography, cryptanalysis, cipher, and cryptographic algorithm. *[Remembering]*	Explain key terms in cryptology, including cryptography, cryptanalysis, cipher, and cryptographic algorithm. *[Understanding]*
IAS-25. Describe basic methods for transforming plaintext into ciphertext. *[Understanding]*	Identify basic methods for transforming plaintext into ciphertext. *[Remembering]*	Describe basic methods for transforming plaintext into ciphertext, such as bit stream and block cipher. *[Understanding]*	Implement basic methods for transforming plaintext into ciphertext, such as bit stream and block cipher. *[Applying]*
IAS-26. Explain the difference between symmetric and asymmetric encryption and how they are collectively used to secure digital communications and e-commerce transactions. *[Understanding]*	Recognize the difference between symmetric and asymmetric encryption. *[Remembering]*	Explain the difference between symmetric and asymmetric encryption and how they are collectively used to secure digital communications and e-commerce transactions. *[Understanding]*	Contrast symmetric and asymmetric encryption and their use in secure digital communications and e-commerce transactions. *[Applying]*
IAS/Web Security KU			
IAS-27. Explain browser and web security model concepts including same-origin policy, web sessions, and secure communication channels. *[Understanding]*	Identify one or more browser and web security model concepts. *[Remembering]*	Explain browser and web security model concepts including same-origin policy, web sessions, and secure communication channels such as TLS. *[Understanding]*	Apply browser and web security model concepts including same-origin policy, web sessions, and secure communication channels such as TLS. *[Applying]*

IAS-28. Investigate common vulnerabilities and attacks in web applications and the coding strategies that are used to mitigate them. *[Applying]*	Describe common vulnerabilities and attacks in web applications and the coding strategies that are used to mitigate them. *[Understanding]*	Investigate common vulnerabilities and attacks in web applications and the coding strategies that are used to mitigate them. *[Applying]*	Examine common vulnerabilities and attacks in web applications and the coding strategies that are used to mitigate them. *[Analyzing]*
IAS/Secure Software Engineering KU			
IAS-29. Write software requirements that include basic security specifications. *[Applying]*	Interpret software requirements that include basic security specifications. *[Understanding]*	Write software requirements that include basic security specifications. *[Applying]*	Evaluate software requirements that include basic security specifications. *[Evaluating]*
IAS-30. Implement a plan to test the security modules in software. *[Applying]*	Summarize a plan to test the security modules in software. *[Understanding]*	Implement a plan to test the security modules in software. *[Applying]*	Integrate a plan to test the security modules in software into the overall test plan. *[Analyzing]*
IAS-31. Investigate security vulnerabilities in a software at the requirement and design phases of the software development life cycle. *[Applying]*	Discuss security vulnerabilities in a software at the requirement and design phases of the software development life cycle. *[Understanding]*	Investigate security vulnerabilities in a software at the requirement and design phases of the software development life cycle. *[Applying]*	Analyze security vulnerabilities in a software at the requirement and design phases of the software development life cycle. *[Analyzing]*

Information Management (IM)

Information Management is primarily concerned with the capture, digitization, representation, organization, transformation, and presentation of information; algorithms for efficient and effective access and updating of stored information; data modeling and abstraction; and physical file storage techniques. Students need the ability to develop conceptual and physical data models; determine which methods and techniques are appropriate for a given problem; and be able to select and implement an appropriate solution that addresses relevant design concerns, including accessibility and usability. (*adapted from cs2013, p. 112*)

Learning Outcome	**Assessment Rubric**		
IM. Information Management KA	**Emerging**	**Developed**	**Highly Developed**
IM/Information Management Concepts KU			
IM-01. Differentiate information from data. *[Understanding]*	Define information and data. *[Remembering]*	Differentiate information from data. *[Understanding]*	Compare information with data. *[Analyzing]*
IM-02. Describe how humans gain access to information and data to support their needs. *[Understanding]*	List ways in which humans gain access to information and data to support their needs. *[Remembering]*	Describe how humans gain access to information and data to support their needs. *[Understanding]*	Illustrate ways in which humans gain access to information and data to support their needs. *[Applying]*
IM-03. Describe the advantages and disadvantages of central organizational control over data. *[Understanding]*	List the advantages and disadvantages of central organizational control over data. *[Remembering]*	Describe the advantages and disadvantages of central organizational control over data. *[Understanding]*	Examine the advantages and disadvantages of central organizational control over data. *[Analyzing]*
IM-04. Describe types of contingency plans including business continuity, disaster recovery and incident response. *[Understanding]*	Recognize contingency plans for various size organizations to include: business continuity, disaster recovery and incident response. *[Remembering]*	Describe types of contingency plans including business continuity, disaster recovery and incident response. *[Understanding]*	Compare contingency plans of various size organizations including business continuity, disaster recovery and incident response. *[Analyzing]*
IM-05. Describe proven techniques to secure data and information. *[Understanding]*	List proven techniques used to secure data and information. *[Remembering]*	Describe specific plans to secure data and information. *[Understanding]*	Implement specific proven techniques to secure data and information. *[Applying]*
IM-06. Investigate	Identify vulnerabilities	Investigate	Examine vulnerabilities

vulnerabilities and failure scenarios in information systems. *[Applying]*	and failure scenarios in information systems. *[Remembering]*	vulnerabilities and failure scenarios in information systems, such as SQL injection and cross-site scripting. *[Applying]*	and failure scenarios in information systems. *[Analyzing]*
IM/Database Systems KU			
IM-07. Explain the characteristics that distinguish the database approach from the approach of programming with data files. *[Understanding]*	Identify the characteristics that distinguish the database approach from the approach of programming with data files. *[Remembering]*	Explain the characteristics that distinguish the database approach from the approach of programming with data files. *[Understanding]*	Contrast the characteristics that distinguish the database approach from the approach of programming with data files. *[Analyzing]*
IM-08. Describe the components of a database system and give examples of their use. *[Understanding]*	Identify the components of a database system. *[Remembering]*	Describe the components of a database system and give examples of their use. *[Understanding]*	Diagram the components of a database system and give examples of their use. *[Applying]*
IM-09. Explain the concept of data independence and its importance in a database system. *[Understanding]*	Define the concept of data independence and its importance in a database system. *[Remembering]*	Explain the concept of data independence and its importance in a database system. *[Understanding]*	Examine the concept of data independence and its importance in a database system. *[Analyzing]*
IM-10. Use SQL or a similar query language to elicit information from a database. *[Applying]*	Describe the process of eliciting information from a database using a query language. *[Understanding]*	Use SQL or a similar query language to elicit information from a database. *[Applying]*	Evaluate a variety of query languages used to elicit information from a database. *[Evaluating]*
IM-11. Describe common security concerns in database management systems. *[Understanding]*	Identify common security concerns in database management systems. *[Remembering]*	Describe common security concerns in database management systems. *[Understanding]*	Distinguish common security concerns in database management systems. *[Analyzing]*
IM-12. Apply security principles to the design and development of database systems. *[Applying]*	Discuss security principles in the design and development of database systems. *[Understanding]*	Apply security principles to the design and development of database systems. *[Applying]*	Assess security principles used in the design and development of database systems. *[Evaluating]*
IM/Data Modeling KU			
IM-13. Contrast appropriate data models, including internal structures, for different types of	Diagram appropriate data models, including internal structures, for	Contrast appropriate data models, including internal structures, for	Evaluate appropriate data models, including internal structures, for

data. *[Analyzing]*	different types of data. *[Applying]*	different types of data. *[Analyzing]*	different types of data. *[Evaluating]*
IM-14. Describe modeling notation concepts and how they are applied. *[Understanding]*	Define modeling notation concepts and how they would be used in a DBMS. *[Remembering]*	Describe modeling notation concepts and how they are applied. *[Understanding]*	Illustrate concepts in data modeling notation and how they might be used. *[Applying]*
IM-15. Diagram a relational data model for a given scenario. *[Applying]*	Describe a relational data model for a given scenario. *[Understanding]*	Diagram a relational data model for a given scenario. *[Applying]*	Analyze a relational data model for a given scenario. *[Analyzing]*
IM-16. Describe the basic concepts of the Object-Oriented (OO) model. *[Understanding]*	Identify the basic concepts of the OO model. *[Remembering]*	Describe the basic concepts of the Object-Oriented (OO) model. *[Understanding]*	Apply the basic concepts of the OO model. *[Applying]*
IM-17. Describe the differences between relational data models and other models such as semi-structured or flexible schema. *[Understanding]*	Recognize the differences between relational data models and other models such as semi-structured or flexible schema such as JSON, NoSQL. *[Remembering]*	Describe the differences between relational data models and other models such as semi-structured or flexible schema such as JSON, NoSQL. *[Understanding]*	Investigate the differences between relational data models and other models such as semi-structured or flexible schema such as JSON, NoSQL. *[Applying]*
IM-18. Explain potential security and privacy concerns during the process of data modeling. *[Understanding]*	Identify potential security and privacy concerns during the process of data modeling. *[Remembering]*	Explain the potential security and privacy concerns during the process of data modeling. *[Understanding]*	Investigate the potential security and privacy concerns during the process of data modeling. *[Applying]*

Networking and Communication (NC)

The Internet, computer networks, cloud computing, and the Internet of Things (IoT) are now ubiquitous and a growing number of computing activities strongly depend on the correct operation of the underlying network. Networks, both fixed and mobile, are a key part of the computing environment of today and tomorrow. Many computing applications that are used today would not be possible without networks. This dependency on the underlying network is likely to increase in the future. (*adapted from cs2013, p. 130*)

Learning Outcome	**Assessment Rubric**		
NC. Networking and Communication KA	**Emerging**	**Developed**	**Highly Developed**
NC/Introduction KU			
NC-01. Explain the basic structure of the Internet. *[Understanding]*	Identify the basic structure of the Internet. *[Remembering]*	Explain the basic structure of the Internet. *[Understanding]*	Diagram the basic structure of the internet. *[Remembering]*
NC-02. Define basic network terminology. *[Remembering]*	Recognize basic network terminology. *[Remembering]*	Define basic network terminology, such as topologies and protocols. *[Remembering]*	Describe basic network terminology. *[Understanding]*
NC-03. Describe the layered structure of a typical networked architecture, including routing and switching. *[Understanding]*	Label the components of a typical networked architecture. *[Remembering]*	Describe the layered structure of a typical networked architecture, including routing and switching. *[Understanding]*	Diagram the layered structure of a typical networked architecture. *[Applying]*
NC-04. Diagram the layers of the OSI model, including associated protocols. *[Applying]*	Describe the layers of the OSI model. *[Understanding]*	Diagram the layers of the OSI model. *[Applying]*	Compare and contrast the layers of the OSI model with the TCP/IP model. *[Analyzing]*
NC/Networked Applications KU			
NC-05. Define the principles behind naming schemes and resource location. [Remembering]	Define at least one principle behind naming schemes and resource location. *[Remembering]*	Define the principles behind naming schemes and resource location. *[Remembering]*	Demonstrate the principles behind naming schemes and resource location. *[Understanding]*
NC-06. Implement a simple distributed network application. *[Applying]*	Identify the components of a simple distributed network application. *[Remembering]*	Implement a simple distributed network application. *[Applying]*	Integrate a simple distributed network application within a server program to exchange data with a

			client. *[Analyzing]*
NC-07. Describe security concerns in designing applications for use over wireless networks. *[Understanding]*	Recognize security concerns in designing applications for use over wireless networks. *[Remembering]*	Describe security concerns in designing applications for use over wireless networks. *[Understanding]*	Illustrate security concerns in designing applications for use over wireless networks. *[Applying]*
NC-08. Discuss secure connectivity among networked applications. *[Understanding]*	Recognize secure connectivity among networked applications. *[Remembering]*	Discuss secure connectivity among networked applications, such as SSH, HTTPS, SFTP. *[Understanding]*	Investigate secure connectivity among networked applications. *[Applying]*
NC-09. Explain the advantages and disadvantages of using virtualized infrastructure in cloud computing. *[Understanding]*	List the advantages and disadvantages of using virtualized infrastructure in cloud computing. *[Remembering]*	Explain the advantages and disadvantages of using virtualized infrastructure in cloud computing. *[Understanding]*	Investigate the advantages and disadvantages of using virtualized infrastructure in cloud computing. *[Applying]*

Operating Systems (OS)

An operating system defines an abstraction of hardware and manages resource sharing among the computer's users. The topics in this area explain the most basic knowledge of operating systems in the sense of interfacing an operating system to networks, teaching the difference between the kernel and user modes, and developing key approaches to operating system design and implementation. The Operating Systems knowledge area is structured to be complementary to the Systems Fundamentals (SF), Networking and Communication (NC), Information Assurance and Security (IAS), and the Parallel and Distributed Computing (PD) knowledge areas. (*adapted from cs2013, p. 135*)

Learning Outcome	**Assessment Rubric**		
OS. Operating Systems KA	**Emerging**	**Developed**	**Highly Developed**
OS/Overview of Operating Systems KU			
OS-01. Explain major objectives, functions and concepts of modern operating systems. *[Understanding]*	List some objectives, functions and concepts of modern operating systems. *[Remembering]*	Explain major objectives, functions, and concepts of modern operating systems. *[Understanding]*	Explain in detail most objectives, functions and concepts of modern operating systems. *[Understanding]*
OS-02. Describe key features of a contemporary operating system. *[Understanding]*	Identify key features of a contemporary operating system. *[Remembering]*	Describe key features of a contemporary operating system, such as scheduling, file systems, user interfaces, and updates. *[Understanding]*	Analyze key features of a contemporary operating system with respect to convenience, efficiency, and the ability to evolve. *[Analyzing]*
OS-03. Differentiate between prevailing types of operating systems. *[Understanding]*	Define prevailing types of operating systems. *[Remembering]*	Differentiate between prevailing types of operating systems, such as networked, mobile, embedded, and real-time. *[Understanding]*	Investigate prevailing types of operating systems. *[Applying]*
OS/Operating System Principles KU			
OS-04. Diagram the interaction of an application with the operating system through an API. *[Applying]*	Summarize the interaction of an application with the operating system through an API. *[Understanding]*	Diagram the interaction of an application with the operating system through an API. *[Applying]*	Test the interaction of an application with the operating system through an API. *[Evaluating]*

OS-05. Describe how computing resources are used by application software and managed by the operating system. *[Understanding]*	Identify which computing resources are used by application software and managed by the operating system. *[Remembering]*	Describe how computing resources are used by application software and managed by the operating system, such as access control. *[Understanding]*	Investigate how computing resources are used by application software and managed by the operating system. *[Applying]*
OS-06. Locate a device list and a driver I/O queue. *[Remembering]*	Recognize a device list. *[Remembering]*	Locate a device list and a driver I/O queue. *[Remembering]*	Explain the use of a device list and driver I/O queue. *[Understanding]*
OS/Concurrency KU			
OS-07. Describe the need for concurrency within the framework of an operating system. *[Understanding]*	Define concurrency within the framework of an operating system. *[Remembering]*	Describe the need for concurrency within the framework of an operating system. *[Understanding]*	Investigate the need for concurrency within the framework of an operating system. *[Applying]*
OS/Memory Management KU			
OS-08. Describe the principles of memory management. *[Understanding]*	Define the principles of memory management. *[Remembering]*	Describe the principles of memory management, such as memory hierarchy and allocation, cost-performance tradeoffs, and caching. *[Understanding]*	Illustrate the principles of memory management. *[Applying]*
OS-09. Describe the principles of virtual memory. *[Understanding]*	List some of the principles of virtual memory. *[Remembering]*	Describe the principles of virtual memory, such as paging and partitioning. *[Understanding]*	Illustrate the principles of virtual memory. *[Applying]*
OS-10. Define the concept of thrashing. *[Remembering]*	Recognize the concept of thrashing. *[Remembering]*	Define the concept of thrashing. *[Remembering]*	Explain the concept of thrashing and how to manage it. *[Understanding]*
OS/Security and Protection KU			
OS-11. Explain the need for protection and security in an Operating System (OS). *[Understanding]*	Identify the need for protection and security in an OS. *[Remembering]*	Explain the need for protection and security in an Operating System. *[Understanding]*	Investigate the need for protection and security in an OS. *[Applying]*
OS-12. Discuss potential threats to operating systems	Identify some potential threats to operating	Discuss potential threats to operating	Investigate potential threats to operating

and the security features designed to guard against them. *[Understanding]*	systems. *[Remembering]*	systems and the security features designed to guard against them. *[Understanding]*	systems and the security features designed to guard against them. *[Applying]*
OS/Virtual Machines KU			
OS-13. Explain the concept of virtualization and how it is realized in hardware and software. *[Understanding]*	Define the concept of virtualization and how it is realized in hardware and software. *[Remembering]*	Explain the concept of virtualization and how it is realized in hardware and software. *[Understanding]*	Investigate the concept of virtualization and how it is realized in hardware and software. *[Applying]*
OS/Device Management KU			
OS-14. Explain the relationship between the physical hardware and the virtual devices maintained by the operating system. *[Understanding]*	Identify the difference between physical hardware and virtual devices. *[Remembering]*	Explain the relationship between the physical hardware and the virtual devices maintained by the operating system. *[Understanding]*	Diagram the relationship between the physical hardware and the virtual devices maintained by the operating system. *[Applying]*

Parallel and Distributed Computing (PD)

The past decade has brought explosive growth in multiprocessor computing, including multi-core processors and distributed data centers. Both parallel and distributed computing entail the logically simultaneous execution of multiple processes, whose operations have the potential to interleave in complex ways. Parallel and distributed computing builds on foundations in many areas, including an understanding of fundamental systems concepts such as concurrency and parallel execution, consistency in state/memory manipulation, and latency. Communication and coordination among processes is rooted in the message-passing and shared-memory models of computing and such algorithmic concepts as atomicity, consensus, and conditional waiting. Distributed systems highlight the problems of security and fault tolerance, emphasize the maintenance of replicated state, and introduce additional issues that bridge computer networking. The Systems Fundamentals knowledge area also contains an introduction to parallelism. (*adapted from cs2013, p. 145*)

Learning Outcome	**Assessment Rubric**		
PD. Parallel and Distributed Computing KA	**Emerging**	**Developed**	**Highly Developed**
PD/Parallelism Fundamentals KU			
PD-01. Differentiate the goal of parallelism from the goal of concurrency. *[Understanding]*	State the goals of parallelism and concurrency. *[Remembering]*	Differentiate the goal of parallelism, such as throughput, from the goal of concurrency, such as controlling access to shared resources. *[Understanding]*	Analyze the goals of parallelism and concurrency. *[Analyzing]*
PD-02. Summarize various programming constructs for synchronization. *[Understanding]*	Identify various programming constructs for synchronization. *[Remembering]*	Summarize various programming constructs for synchronization. *[Understanding]*	Implement various programming constructs for synchronization. *[Applying]*
PD-03. Differentiate data races from higher level races. *[Understanding]*	Recognize data races from higher level races. *[Remembering]*	Differentiate data races from higher level races. *[Understanding]*	Distinguish data races from higher level races. *[Analyzing]*
PD/Communication and Coordination KU			
PD-04. Explain mutual exclusion as a way to avoid a given race condition. *[Applying]*	Define mutual exclusion. *[Remembering]*	Explain mutual exclusion as a way to avoid a given race condition. *[Understanding]*	Execute mutual exclusion to avoid a given race condition. *[Applying]*

Programming Languages (PL)

Programming languages are the medium through which programmers precisely describe concepts, formulate algorithms, and reason about solutions. A computer scientist will work with many different languages, separately or together. Software developers must understand the programming models underlying different languages and make informed design choices in languages supporting multiple complementary approaches. Computer scientists will often need to learn new languages and programming constructs, and must understand the principles underlying how programming language features are defined, composed, and implemented. The effective use of programming languages, and appreciation of their limitations, also requires a basic knowledge of programming language translation and static program analysis, as well as run-time components and the memory management hierarchy. (*adapted from cs2013, p. 155*)

Learning Outcome	**Assessment Rubric**		
PL. Programming Languages KA	**Emerging**	**Developed**	**Highly Developed**
PL/Object-Oriented Programming KU			
PL-01. Implement a simple secure class hierarchy. *[Applying]*	Describe a simple secure class hierarchy, including superclasses and subclasses, using encapsulation, abstraction, inheritance, and polymorphism. *[Understanding]*	Implement a simple secure class hierarchy, including superclasses and subclasses, using encapsulation, abstraction, inheritance, and polymorphism. *[Applying]*	Implement a complex secure class hierarchy, including superclasses and subclasses, using encapsulation, abstraction, inheritance, and polymorphism. *[Applying]*
PL-02. Use control flow in a program using dynamic dispatch. *[Applying]*	Describe the best ways to use flow control in a program using dynamic dispatch. *[Understanding]*	Use control flow in a program using dynamic dispatch. *[Applying]*	Contrast control flow using a static environment vs a dynamic environment. *[Analyzing]*
PL-03. Use access modifiers to secure class data. *[Applying]*	Describe access modifiers to secure class data such as private and protected. *[Understanding]*	Use access modifiers to secure class data such as private and protected. *[Applying]*	Compare the effect of access modifiers to secure data. *[Analyzing]*
PL-04. Describe the tenets of OOP, including encapsulation, abstraction, inheritance, and polymorphism and how they impact/influence security. *[Understanding]*	**Recognize design patterns in the software lifecycle components.** *[Remembering]*	**Describe the tenets of OOP, including encapsulation, abstraction, inheritance, and polymorphism and how they**	**Compare design patterns and their impact to achieve reliability and security in the software development.** *[Analyzing]*

		impact/influence security. *[Understanding]*	
PL/Functional Programming KU			
PL-05. Write basic algorithms that avoid assigning to mutable state or considering reference equality. *[Applying]*	Discuss how in functional languages, rather than mutating the state of objects, the objects are simply returned with the desired reflected changes. *[Understanding]*	Write basic algorithms that avoid assigning to mutable state or considering reference equality. *[Applying]*	Evaluate efficiency by comparing algorithms that avoid assigning to mutable state vs modifying data structures values. *[Evaluating]*
PL-06. Compare and contrast the procedural/functional approach and the object-oriented approach. *[Analyzing]*	Differentiate between different programming paradigms by mention advantages and disadvantages of each paradigm. *[Understanding]*	Compare and contrast the procedural/functional approach and the object-oriented approach. *[Analyzing]*	Apply a function for a given operation with the function body providing a case for each data variant. *[Applying]*
PL-07. Write useful functions that take and return other functions. *[Applying]*	Explain what Lambda calculus is and how this principle is used in computer programming. *[Understanding]*	Write useful functions that take and return other functions. *[Applying]*	Test several functions with different definitions to produce a desired output based on returning other functions. *[Evaluating]*
PL/Event-Driven and Reactive Programming KU			
PL-08. Describe event handlers for use in reactive systems. *[Understanding]*	Define the purpose of event handlers in reactive system such as GUIs. *[Remembering]*	Describe event handlers for use in reactive systems, such as GUIs. *[Understanding]*	Write event handlers for use in reactive systems. *[Applying]*
PL-09. Explain why an event-driven programming style is natural in domains where programs react to external eve9nts. *[Remembering]*	Describe an advantage of having an event-driven programming style vs a pre-defined programming style. *[Understanding]*	Explain why an event-driven programming style is natural in domains where programs react to external events. *[Remembering]*	Use an event-driven style to generate outcomes that can be evaluate them. *[Applying]*
PL-10. Describe potential security vulnerabilities in event-driven GUI applications. *[Understanding]*	Identify potential security vulnerabilities in event-driven GUI applications. *[Remembering]*	Describe potential security vulnerabilities in event-driven GUI applications, such as injection-based attacks. *[Understanding]*	Illustrate potential security vulnerabilities in event-driven GUI applications.. *[Applying]*

PL/Basic Type Systems KU			
PL-11. Describe strong-type and weak-type languages and potential errors that are detected. *[Understanding]*	List a set of strong-type and weak-type languages. *[Remembering]*	Describe strong-type and weak-type languages and potential errors that are detected. *[Understanding]*	Compare strong-type and weak-type languages and potential errors that are detected. *[Analyzing]*
PL-12. Explain the security implications of a type-safe language for software development. *[Understanding]*	Recognize the security advantages of a type-safe language for software development. *[Remembering]*	Explain the security advantages of a type-safe language for software development. *[Understanding]*	Examine the security advantages of a type-safe language for software development. *[Analyzing]*

Software Development Fundamentals (SDF)

Fluency in the process of software development is a prerequisite to the study of most of computer science. In order to use computers to solve problems effectively, students must be competent at reading and writing programs in multiple programming languages. Beyond programming skills, however, they must be able to design and analyze algorithms, select appropriate paradigms, and utilize modern development and testing tools. This knowledge area brings together those fundamental concepts and skills related to the software development process. As such, it provides a foundation for other software-oriented knowledge areas, most notably Programming Languages, Algorithms and Complexity, and Software Engineering. (*adapted from CS2013, p. 167*)

Learning Outcome	**Assessment Rubric**		
SDF. Software Development Fundamentals KA	**Emerging**	**Developed**	**Highly Developed**
SDF/Algorithms and Design KU			
SDF-01. Discuss the importance of algorithms in the problem-solving process. *[Understanding]*	Recognize the importance of using algorithms in the problem-solving process. *[Remembering]*	Discuss the importance of algorithms in the problem-solving process. *[Understanding]*	Examine the importance of algorithms in the problem-solving process. *[Analyzing]*
SDF-02. Implement an algorithm in a programming language to solve problems. *[Applying]*	Explain the steps of an algorithm for solving a problem. *[Understanding]*	Implement an algorithm in a programming language to solve problems. *[Applying]*	Devise an algorithm in a programming language to solve problems. *[Creating]*
SDF-03. Use the techniques of decomposition to break a program into smaller pieces. *[Applying]*	Define decomposition with regard to its use in computer science. *[Remembering]*	Use the techniques of decomposition to break a program up into smaller pieces. *[Applying]*	Analyze code to see how decomposition techniques were used. *[Analyzing]*
SDF-04. Differentiate strengths and weaknesses among multiple algorithms for a problem. *[Understanding]*	Identify some strengths and weaknesses of multiple algorithms for a problem. *[Remembering]*	Differentiate strengths and weaknesses among multiple algorithms for a problem. *[Understanding]*	Analyze strengths and weaknesses of multiple algorithms for a problem. *[Analyzing]*
SDF-05. Explain the differences between brute-force and divide-and-conquer algorithms in relation to security objectives. *[Understanding]*	List the differences between brute-force and divide-and-conquer algorithms in relation to security objectives. *[Remembering]*	Explain the differences between brute-force and divide-and-conquer algorithms in relation to security objectives, such as cracking a	Compare brute-force and divide-and-conquer algorithms in achieving security objectives. *[Analyzing]*

		password. *[Understanding]*	
SDF/Fundamental Programming Concepts KU			
SDF-06. Write programs that use abstract data types (ADTs). *[Applying]*	Describe abstract data types (ADTs). *[Understanding]*	Write programs that use abstract data types (ADTs). *[Applying]*	Write complex programs that use abstract data types (ADTs). *[Applying]*
SDF-07. Investigate potential vulnerabilities in programming code. *[Applying]*	Identify potential vulnerabilities in programming code. *[Remembering]*	Investigate potential vulnerabilities in programming code. *[Applying]*	Choose a solution to mitigate vulnerabilities in programming code. *[Evaluating]*
SDF-08. Write programs which use defensive programming techniques, including input validation, type checking, and protection against buffer overflow. *[Applying]*	Describe defensive programming techniques. [*Understanding*]	Write programs which use defensive programming techniques, including input validation, type checking, and protection against buffer overflow. *[Applying]*	Write complex programs which use defensive programming techniques, including input validation, type checking, and protection against buffer overflow. *[Applying]*
SDF-09. Write programs that use variables, expressions, assignments, and I/O. *[Applying]*	Describe programs that use variables, expressions, assignments, and I/O. *[Understanding]*	Write programs that use variables, expressions, assignments, and I/O. *[Applying]*	Write complex programs that use variables, expressions, assignments, and I/O. *[Applying]*
SDF-10. Write programs that use primitive data types in a programming language. *[Applying]*	Describe primitive data types. [*Understanding*]	Write programs that use primitive data types in a programming language. [*Applying*]	Compare characteristics of primitive data types. [*Analyzing*]
SDF-11. Write programs that use sequence, conditional, and iterative control structures. *[Applying]*	Describe sequence, conditional, and iterative control structures. [*Understanding*]	Write programs that use sequence, conditional, and iterative control structures. [*Applying*]	Write complex programs that use sequence, conditional, and iterative control structures. [*Applying*]
SDF-12. Write programs that use functions, parameters, and return values. *[Applying]*	Describe functions, parameters, and return values. [*Understanding*]	Write programs that use functions, parameters, and return values. [*Applying*]	Write complex programs that use functions, parameters, and return values. [*Applying*]
SDF-13. Write a program that uses persistence to save data across multiple executions.	Describe the concept of persistence.	Write a program that uses persistence to save data across	Write a complex program that uses persistence to save

[Applying]	[*Understanding*]	multiple executions. *[Applying]*	data across multiple executions. *[Applying]*
SDF-14. Write a program that uses recursion. *[Applying]*	Describe recursion by tracing the stack. [*Understanding*]	Write a program that uses recursion. [*Applying*]	Compare types of recursion techniques, such as binary, tail, and natural recursion. [*Analyzing*]
SDF/Fundamental Data Structures KU			
SDF-15. Write programs that implement each of the following data structures: lists, stacks, queues, hash tables, and trees. *[Applying]*	Describe each of the following data structures: lists, stacks, queues, hash tables, and trees. *[Understanding]*	Write programs that implement each of the following data structures: lists, stacks, queues, hash tables, and trees. *[Applying]*	Write complex programs that implement each of the following data structures: lists, stacks, queues, hash tables, and trees. *[Applying]*
SDF-16. Compare the efficiency of basic operations across various data structures. *[Analyzing]*	Investigate the efficiency of basic operations across various data structures. *[Applying]*	Compare the efficiency of basic operations, such as insertion and deletion, across various data structures. [*Analyzing*]	Critique the efficiency of basic operations across various data structures. [*Evaluating]*
SDF-17. Implement references in programming code which allow data to be accessed in multiple ways. *[Applying]*	Describe how references in programming code allow data to be accessed in multiple ways. *[Understanding]*	Implement references in programming code which allow data to be accessed in multiple ways. *[Applying]*	Evaluate the use of references in programming code. *[Evaluating]*
SDF/Development Methods KU			
SDF-18. Describe common coding errors that expose security vulnerabilities. *[Understanding]*	List common coding errors that expose security vulnerabilities. *[Remembering]*	Describe common coding errors that expose security vulnerabilities, such as buffer overflows, integer errors, and memory leaks. *[Understanding]*	Investigate common coding errors that introduce security vulnerabilities and the associated techniques for securing the code. *[Applying]*
SDF-19. Implement simple refactoring within given program components. *[Applying]*	List opportunities within given program components for simple refactoring. *[Remembering]*	Implement simple refactoring within given program components. *[Applying]*	Implement refactoring within complex program components. *[Applying]*
SDF-20. Describe programming by contract and the role of	List concepts associated with	Describe programming by	Implement a program utilizing preconditions,

preconditions, postconditions, and invariants. *[Understanding]*	programming by contract. *[Remembering]*	contract and the role of preconditions, postconditions, and invariants. *[Understanding]*	postconditions, and invariants. *[Applying]*
SDF-21. Apply a variety of strategies to test and debug programs. *[Applying]*	List strategies to test and debug programs. *[Remembering]*	Apply a variety of strategies to test and debug programs, such as unit testing and test-case generation. *[Applying]*	Analyze a variety of strategies to test and debug programs. *[Analyzing]*
SDF-22. Use an integrated development environment (IDE) to create, execute, and debug secure programs. *[Applying]*	Discuss the benefits of using an integrated development environment (IDE) to create, execute, and debug secure programs. *[Understanding]*	Use an integrated development environment (IDE) to create, execute, and debug secure programs. *[Applying]*	Compare integrated development environments (IDEs) for a given programming language. *[Analyzing]*
SDF-23. Use standard libraries for a given programming language. *[Applying]*	List the standard libraries for a given programming. *[Remembering]*	Use standard libraries for a given programming language. [*Applying*]	Choose appropriate components from standard libraries to solve a given problem. [*Evaluating*]
SDF-24. Apply consistent documentation and program style standards. *[Applying]*	Explain the reasons for using consistent documentation and program style standards. [*Understanding*]	Apply consistent documentation and program style standards. [*Applying*]	Assess documentation and program style in a given program. [*Evaluating*]
SDF-25. Carry out a code review on a program component using a provided security checklist. *[Applying]*	Explain the process of a code review. [*Understanding*]	Carry out a code review on a program component using a provided security checklist. [*Applying*]	Organize a team code review on a program component using a provided security checklist. [*Analyzing*]

Software Engineering (SE)

Software engineering is the discipline concerned with the application of theory, knowledge, and practice to effectively and efficiently build reliable software systems that satisfy the requirements of customers and users. This discipline is applicable to small, medium, and large-scale systems. It encompasses all phases of the lifecycle of a software system, including requirements elicitation, analysis and specification; design; construction; verification and validation; deployment; and operation and maintenance. Whether small or large, following a traditional plan-driven development process, an agile approach, or some other method, software engineering is concerned with the best way to build good software systems.

Software engineering uses engineering methods, processes, techniques, and measurements. It benefits from the use of tools for managing software development; analyzing and modeling software artifacts; assessing and controlling quality; and for ensuring a disciplined, controlled approach to software evolution and reuse. The software engineering toolbox has evolved over the years. For instance, the use of contracts, with requires and ensure clauses and class invariants, is one good practice that has become more common. Software development, which can involve an individual developer or a team or teams of developers, requires choosing the most appropriate tools, methods, and approaches for a given development environment. Practicing software engineers have to select and apply appropriate techniques and practices to a given development effort in order to maximize value. (*adapted from cs2013, p. 172*)

Learning Outcome	**Assessment Rubric**		
SE. Software Engineering KA	**Emerging**	**Developed**	**Highly Developed**
SE/Software Processes KU			
SE-01. Describe how software interacts with various systems, including information management, embedded, process control, and communications systems. *[Understanding]*	Recognize how software interacts with various systems, including information management, embedded, process control, and communications systems. *[Remembering]*	Describe how software interacts with various systems including information management, embedded, process control, and communications systems. *[Understanding]*	Diagram how software interacts with various systems, including information management, embedded, process control, and communications systems. *[Applying]*
SE-02. Describe the relative advantages and disadvantages among several software process models. *[Understanding]*	Identify some advantages and disadvantages among several software process models. *[Remembering]*	Describe the relative advantages and disadvantages among several software process models, such as waterfall, iterative, and agile. *[Understanding]*	Contrast the relative advantages and disadvantages among several process models. *[Analyzing]*
SE-03. Describe the phases of secure software development	List the phases of secure software	Describe the phases of secure software	Perform the phases of secure software

lifecycle (SecSDLC). *[Understanding]*	development lifecycle (SecSDLC). *[Remembering]*	development lifecycle (SecSDLC). *[Understanding]*	development lifecycle (SecSDLC). *[Applying]*
SE/Software Project Management KU			
SE-04. Illustrate common behaviors that contribute to the effective functioning of a team. *[Applying]*	List common behaviors that contribute to the effective functioning of a team. *[Remembering]*	Illustrate common behaviors that contribute to the effective functioning of a team, such as good communication skills. *[Applying]*	Examine common behaviors that contribute to the effective functioning of a team. *[Analyzing]*
SE-05. Explain the risks in using third-party code. *[Understanding]*	List the risks using third-party code. *[Remembering]*	Explain the risks using third-party code, such as web-based compiler services. *[Understanding]*	Illustrate the risks using third-party code. *[Applying]*
SE/Tools and Environments KU			
SE-06. Describe how version control can be used to help with software release management. *[Understanding]*	Identify version control technologies that can be used for software release management. *[Remembering]*	Describe how version control can be used to help with software release management, such as using revision software control services. *[Understanding]*	Use version control for software release management. *[Applying]*
SE-07. Describe a set of development tools for software systems *[Understanding]*	Recognize a set of development tools for software systems. *[Remembering]*	Describe a set of development tools for software systems, such as requirements tracking, modeling, automation, and testing. *[Understanding]*	Investigate a set of development tools for software systems. *[Applying]*
SE/Requirements Engineering KU			
SE-08. Describe a key feature that is required for a given software system. *[Understanding]*	List a key feature that is required for a given software system. *[Remembering]*	Describe a key feature that is required for a given software system. *[Understanding]*	Write the requirements for a key feature for a given software system. *[Applying]*
SE-09. Describe how the requirements engineering process supports the elicitation and validation of behavioral requirements. *[Understanding]*	Define the requirements engineering in the context of behavioral requirements. *[Remembering]*	Describe how the requirements engineering process supports the elicitation and validation of behavioral requirements. *[Understanding]*	Develop software requirements based on the elicitation and validation of behavioral requirements. *[Creating]*

SE-10. Interpret a given requirements model for a simple software system. *[Understanding]*	Recognize a given requirements model for a simple software system. *[Remembering]*	Interpret a given requirements model for a simple software system. *[Understanding]*	Produce a requirements model for a simple software system. *[Applying]*
SE/Software Design KU			
SE-11. Describe different software design principles. *[Understanding]*	Recognize different software design principles. *[Remembering]*	Describe different software design principles, such as architectural and detailed design, separation of concerns, information hiding, coupling and cohesion, and reuse of standard structures. *[Understanding]*	Illustrate different software design principles. *[Applying]*
SE-12. Analyze an existing software design to improve its security. *[Analyzing]*	Identify possible stages of software design that may introduce a security vulnerability. *[Remembering]*	Analyze an existing software design to improve its security. *[Analyzing]*	Debate whether a proposed solution/patch to the design can fix the vulnerability in a viable and effective way. *[Evaluating]*
SE-13. Describe the cost and tradeoffs associated with designing security into software. *[Understanding]*	Recognize situations where security designs are effectively applied in software. *[Remembering]*	Describe the cost and tradeoffs associated with designing security into software. *[Understanding]*	Compare security software designs and associated costs and tradeoffs. *[Analyzing]*
SE-14. Explain first principles of software security. *[Understanding]*	List some first principles of software security. *[Remembering]*	Explain first principles of software security, such as least privilege, information hiding, and simplicity. *[Understanding]*	Illustrate first principles of software security. *[Applying]*
SE/Software Construction KU			
SE-15. Implement a defined coding standard in a small software project. *[Applying]*	Select a defined coding standard in a small software project. *[Remembering]*	Implement a defined coding standard in a small software project. *[Applying]*	Justify the reason for using a given coding standard. *[Evaluating]*
SE/Software Verification and Validation KU			
SE-16. Differentiate between program validation and verification. *[Understanding]*	Recognize program components which do not meet specified requirements. *[Remembering]*	Differentiate between program validation and verification. *[Understanding]*	Perform software validation and verification for a given piece of code. *[Applying]*

SE-17. Describe different types and levels of testing. *[Understanding]*	List different types and levels of testing. *[Remembering]*	Describe different types and levels of testing, such as unit testing, system testing, integration testing, and interface usability. *[Understanding]*	Illustrate different types and levels of testing for a given program. *[Applying]*
SE-18. Describe security tests performed during software development. *[Understanding]*	Recognize potential security vulnerabilities by testing given specifications. *[Remembering]*	Describe security tests performed during software development. *[Understanding]*	Perform security tests for a given program. *[Analyzing]*

Systems Fundamentals (SF)

The underlying hardware and software infrastructure upon which applications are constructed is collectively described by the term "computer systems." Computer systems broadly span the sub-disciplines of operating systems, parallel and distributed systems, communications networks, and computer architecture. Traditionally, these areas are taught in a non-integrated way through independent courses. However these sub-disciplines increasingly share important common fundamental concepts within their respective cores. These concepts include computational paradigms, parallelism, cross-layer communications, state and state transition, resource allocation and scheduling, and so on. The Systems Fundamentals Knowledge Area is designed to present an integrative view of these fundamental concepts in a unified albeit simplified fashion, providing a common foundation for the different specialized mechanisms and policies appropriate to the particular domain area. (*adapted from cs2013, p. 186*)

Learning Outcome	**Assessment Rubric**		
SF. System Fundamentals KA	**Emerging**	**Developed**	**Highly Developed**
SF/Computational Paradigms KU			
SF-01. List commonly encountered patterns of how computations are organized. *[Remembering]*	Recognize some patterns of how computations are organized. *[Remembering]*	List commonly encountered patterns of how computations are organized. *[Remembering]*	Explain commonly encountered patterns of how computations are organized. *[Understanding]*
SF-02. Identify the basic building blocks of computers and their role in the historical development of computer architecture. *[Remembering]*	Identify some of the basic building blocks of computers. *[Remembering]*	Identify the basic building blocks of computers and their role in the historical development of computer architecture. *[Remembering]*	Discuss the basic building blocks of computers and their role in the historical development of computer architecture. *[Remembering]*
SF-03. Discuss the differences between single thread and multiple thread, single server and multiple server models. *[Understanding]*	Identify some differences between single thread and multiple thread, single server and multiple server models. *[Remembering]*	Discuss the differences between single thread and multiple thread, single server and multiple server models. *[Understanding]*	Illustrate the differences between single thread and multiple thread, single server and multiple server models. *[Applying]*
SF-04. Illustrate performance of simple sequential and parallel versions of a program with different problem sizes. *[Applying]*	Discuss the general performance of simple sequential and parallel versions of a program. *[Understanding]*	Illustrate performance of simple sequential and parallel versions of a program with different problem sizes. *[Applying]*	Compare performance of sequential and parallel versions of a program with different problem

			sizes. *[Analyzing]*
SF-05. Recognize security implications related to emerging computational paradigms. *[Remembering]*	List one or more security implications related to computational paradigms. *[Remembering]*	Recognize security implications related to emerging computational paradigms, such as quantum computing and biological computing. *[Remembering]*	Discuss security implications related to emerging computational paradigms. *[Understanding]*
SF/Cross-Layer Communications KU			
SF-06. Describe how computing systems are constructed of layers upon layers. *[Understanding]*	Recognize that computing systems are constructed of layers upon layers. *[Remembering]*	Describe how computing systems are constructed of layers upon layers, such as separation of concerns, well-defined interfaces, and abstraction. *[Understanding]*	Diagram a computing systems constructed of layers upon layers. *[Applying]*
SF-07. Implement a simple program using methods of layering. *[Applying]*	Explain a simple program that uses methods of layering. *[Understanding]*	Implement a simple program using methods of layering, such as error detection, recovery and status across layers. *[Applying]*	Implement a complex program using methods of layering. *[Applying]*
SF-08. Investigate defects in a layered program using tools for program tracing, single stepping, and debugging. *[Applying]*	Demonstrate defects in a layered program using tools for program tracing, single stepping, and debugging. *[Understanding]*	Investigate defects in a layered program using tools for program tracing, single stepping, and debugging. *[Applying]*	Categorize defects by security risk in a layered program using tools for program tracing, single stepping, and debugging. *[Analyzing]*
SF/Parallelism KU			
SF-09. For a given program, differentiate between its sequential and parallel execution. *[Understanding]*	For a given program, identify at least one difference between its sequential and parallel execution. *[Remembering]*	For given program, differentiate between its sequential and parallel execution, such as, with respect to performance implications. *[Understanding]*	For given program, investigate its sequential and parallel execution including performance implications. *[Applying]*
SF-10. Summarize the differences among the concepts of Instruction Parallelism, Data Parallelism, Thread Parallelism/Multitasking, Task/Request Parallelism.	Define the concepts of Instruction Parallelism, Data Parallelism, Thread Parallelism/ Multitasking,	Summarize the differences among the concepts of Instruction Parallelism, Data Parallelism, Thread Parallelism/Multitasking,	Investigate the differences among the concepts of Instruction Parallelism, Data Parallelism, Thread

[Understanding]	Task/Request Parallelism. *[Remembering]*	Task/Request Parallelism. *[Understanding]*	Parallelism/Multitasking, Task/Request Parallelism. *[Applying]*

Social Issues and Professional Practice (SP)

As technological advances continue to significantly impact the way we live and work, the critical importance of social issues and professional practice continues to increase. While technical issues are central to the computing curriculum, they do not constitute a complete educational program in the field. Students must also be exposed to the larger societal context of computing to develop an understanding of the relevant social, ethical, legal and professional issues.

Students also need to develop the ability to ask serious questions about the social impact of computing and to evaluate proposed answers to those questions. Future practitioners must be able to anticipate the impact of introducing a given product into a given environment. Will that product enhance or degrade the quality of life? What will the impact be upon individuals, groups, and institutions? Finally, students need to be aware of the basic legal rights of software and hardware vendors and users, and they also need to appreciate the ethical values that are the basis for those rights. Future practitioners must understand the responsibility that they will bear, and the possible consequences of failure. They must understand their own limitations as well as the limitations of their tools. All practitioners must make a long-term commitment to remaining current in their chosen specialties and in the discipline of computing as a whole.

The application of ethical analysis and reasoning underlies every subsection of this Social Issues and Professional Practices knowledge area in computing. The ACM Code of Ethics and Professional Conduct (http://www.acm.org/about/code-of-ethics) provides guidelines that serve as the basis for the conduct of our professional work. The General Moral Imperatives provide an understanding of our commitment to personal responsibility, professional conduct, and our leadership roles. Computing faculty who are unfamiliar with the content and/or pedagogy of applied ethics are urged to take advantage of the considerable resources from ACM and its Special Interest Group on Computers and Society (SIGCAS). (*adapted from cs2013, p. 192*)

Learning Outcome	**Assessment Rubric**		
SP. Social Issues and Professional Practice KA	**Emerging**	**Developed**	**Highly Developed**
SP/Social Context KU			
SP-01. Investigate positive and negative ways in which computer technology alters modes of information exchange and social interaction. *[Applying]*	Identify ways in which computer technology affects social interaction. *[Remembering]*	Investigate positive and negative ways in which computer technology, such as networks, mobile computing, cloud	Analyze positive and negative ways in which computer technology alters modes of information exchange and social

		computing, alters modes of information exchange and social interaction. *[Applying]*	interactions. *[Analyzing]*
SP-02. Interpret developers' assumptions and values embedded in hardware and software design, especially as they pertain to usability for diverse populations. *[Understanding]*	Identify developers' assumptions and values embedded in hardware and software design, especially as they pertain to underrepresented populations and the disabled. *[Remembering]*	Interpret developers' assumptions and values embedded in hardware and software design, especially as they pertain to usability for diverse populations including under-represented populations and the disabled. *[Understanding]*	Contrast developers' assumptions and values embedded in hardware and software design, especially as they pertain to usability for diverse populations including under-represented populations and the disabled. *[Analyzing]*
SP-03. Describe the impact of underrepresented, diverse populations in the computing profession. *[Understanding]*	Identify how the underrepresentation of diverse populations impacts the computing profession, such as: industry culture, product diversity. *[Remembering]*	Describe the impact of underrepresented, diverse populations in the computing profession, such as industry culture and product development. *[Understanding]*	Analyze the impact of underrepresented, diverse populations in the computing profession, such as industry culture and product development. *[Analyzing]*
SP-04. Describe potential social engineering attacks and the bad actors who might perform them. *[Understanding]*	Identify potential social engineering attacks and the bad actors who might perform them. *[Remembering]*	Describe potential social engineering attacks and the bad actors who might perform them. *[Understanding]*	Analyze the impact and likelihood of potential social engineering attacks. *[Analyzing]*
SP/Analytical Tools KU			
SP-05. Describe stakeholder positions in a given scenario. *[Understanding]*	Define stakeholder positions in a given scenario. *[Remembering]*	Describe stakeholder positions in a given scenario. *[Understanding]*	Illustrate stakeholder positions in a given scenario. *[Applying]*
SP-06. Discuss ethical/social tradeoffs in technical decisions. *[Understanding]*	Identify ethical/social tradeoffs in technical decisions. *[Remembering]*	Discuss ethical/social tradeoffs in technical decisions. *[Understanding]*	Examine ethical/social tradeoffs in technical decisions. *[Analyzing]*
SP/Professional Ethics KU			
SP-07. Investigate various types of ethical issues and dilemmas in enterprise computing. *[Applying]*	Identify various types of ethical issues and dilemmas in enterprise computing. *[Remembering]*	Investigate various types of ethical issues and dilemmas in enterprise computing. *[Applying]*	Debate various types of ethical issues and dilemmas in enterprise computing. *[Evaluating]*

SP-08. Investigate recent and historical ethical violations related to unlawful access and software piracy. *[Applying]*	Identify recent and historical ethical violations related to unlawful access and software piracy. *[Remembering]*	Investigate recent and historical ethical violations related to unlawful access and software piracy. *[Applying]*	Assess the consequences of ethical violations related to unlawful access and software piracy. *[Evaluating]*
SP-09. Analyze the impact of Acceptable Use Policies and online codes of conduct from professional societies on employee behavior and ethical choices. *[Analyzing]*	Interpret the most commonly listed items in an Acceptable Use Policy and an online code of conduct as they relate to ethical choices. *[Understanding]*	Analyze the impact of Acceptable Use Policies and online codes of conduct from professional societies on employee behavior and ethical choices. *[Analyzing]*	Critique Acceptable Use Policies and online codes of conduct for their relevance to the current professional environment. *[Evaluating]*
SP/Intellectual Property KU			
SP-10. Differentiate the terms intellectual property, fair-use, copyright, patent, trademark, and plagiarism. *[Understanding]*	Define the terms intellectual property, fair-use, copyright, and plagiarism. Give examples of each. State the plagiarism policy at your school. *[Remembering]*	Differentiate the terms intellectual property, fair-use, copyright, patent, trademark, and plagiarism. *[Understanding]*	Investigate ethics violations related to intellectual property rights, fair-use, copyright, patents, trademarks, and plagiarism. *[Applying]*
SP-11. Investigate the key pieces of legislation related to fair-use, plagiarism, and intellectual property rights. *[Applying]*	Identify the key pieces of legislation related to fair-use, plagiarism, and intellectual property copyrights. *[Remembering]*	Investigate the key pieces of legislation related to fair-use, plagiarism, and intellectual property rights. *[Applying]*	Debate the effectiveness of legislation related to the fair-use doctrine, plagiarism, intellectual property copyright protections in the global marketplace and economy. *[Evaluating]*
SP-12. Summarize laws, both national and international, related to software patents, trademarks, and copyrights. *[Understanding]*	Define the process of obtaining a software/ hardware patent and the laws that protect the patent. *[Remembering]*	Summarize laws, both national and international, related to software patents, trademarks, and copyrights. *[Understanding]*	Analyze the process of obtaining a software/hardware patent and the laws that protect the patent by referencing key cases such as Apple v Samsung. *[Analyzing]*
SP/Privacy and Civil Liberties KU			
SP-13. Investigate threats to privacy rights in personally identifiable information (PII). *[Applying]*	Identify solutions for privacy threats to personally identifiable information. *[Remembering]*	Investigate threats to privacy rights in personally identifiable information (PII). *[Applying]*	Analyze solutions for privacy threats to personally identifiable Information. *[Analyzing]*

SP-14. Investigate the role of data collection in the implementation of pervasive surveillance systems. *[Applying]*	Identify the types of data collected in the implementation of pervasive surveillance systems. *[Remembering]*	Investigate the role of data collection in the implementation of pervasive surveillance systems, such as RFID, face recognition, and mobile computing. *[Applying]*	Assess the role of data collection in the implementation of pervasive surveillance systems. *[Evaluating]*
SP-15. Investigate technological solutions to privacy concerns. *[Applying]*	Identify technological solutions to privacy concerns. *[Remembering]*	Investigate technological solutions to privacy concerns. *[Applying]*	Debate the impact of technological solutions to privacy problems. *[Evaluating]*
SP/Professional Communication KU			
SP-16. Use effective oral, written, and visual communication techniques in the computing profession. *[Applying]*	Identify effective oral, written, and visual communication techniques for the computing profession. *[Remembering]*	Use effective oral, written, and visual communication techniques in the computing profession. *[Applying]*	Evaluate effective oral, written, and visual communication techniques for the computing profession. *[Evaluating]*
SP-17. Interpret the impact of both verbal and nonverbal cues during face-to-face communications. *[Understanding]*	List the various ways in which verbal and nonverbal cues can impact the message being delivered during face-to-face communications. *[Remembering]*	Interpret the impact of both verbal and nonverbal cues during face-to-face communications. *[Understanding]*	Distinguish the impact of both verbal and nonverbal cues during face-to-face communications. *[Analyzing]*
SP-18. Demonstrate established communication theories as they apply to technical writing and technical communication. *[Understanding]*	Identify the communication theories used in technical writing and technical communication. *[Remembering]*	Demonstrate established communication theories used in technical writing and technical communication. *[Understanding]*	Apply established communication theories in technical writing and technical communication. *[Applying]*
SP/Sustainability KU			
SP-19. Describe the various ways in which computing or use of e-resources impacts the economy, the society, and the environment around us. *[Remembering]*	List the various ways in which computing or use of e-resources impact the economy, the society, and the environment around us. *[Remembering]*	Describe the various ways in which computing or use of e-resources impact the economy, the society, and the environment around us. *[Understanding]*	Analyze various ways in which computing or use of e-resources impact the economy, the society, and the environment around us. *[Analyzing]*
SP-20. Summarize case studies related to sustainable computing efforts. *[Understanding]*	Identify sustainable computing efforts in a case study. *[Remembering]*	Summarize case studies related to sustainable computing efforts.	Investigate solutions for the problems listed in a case study related to sustainable

		[Understanding]	computing. *[Applying]*
SP/Security Policies, Laws and Computer Crime KU			
SP-21. Summarize examples of computer crimes and social engineering incidents with societal impact. *[Understanding]*	List examples of computer crimes and social engineering incidents with societal impact. *[Remembering]*	Summarize examples of computer crimes and social engineering incidents with societal impact. *[Understanding]*	Investigate recent examples of computer crimes and social engineering incidents with societal impact. *[Applying]*
SP-22. Interpret laws that apply to computer crimes. *[Understanding]*	Identify laws that apply to computer crimes. *[Remembering]*	Interpret laws that apply to computer crimes. *[Understanding]*	Critique laws that apply to computer crimes. *[Evaluating]*
SP-23. Describe the motivation and ramifications of cyber terrorism and criminal hacking. *[Understanding]*	Recognize the motivation and ramifications of cyber terrorism and criminal hacking. *[Remembering]*	Describe the motivation and ramifications of cyber terrorism and criminal hacking. *[Understanding]*	Evaluate the motivation and ramifications of cyber terrorism and criminal hacking. *[Evaluating]*
SP-24. Examine the ethical and legal issues surrounding the misuse of access and various breaches in security. *[Analyzing]*	Discuss the ethical and legal issues surrounding the misuse of access and various breaches in security. *[Understanding]*	Examine the ethical and legal issues surrounding the misuse of access and various breaches in security. *[Analyzing]*	Debate the ethical and legal issues surrounding the misuse of access and various breaches in security *[Evaluating]*.
SP-25. Write a security policy, which includes procedures for managing passwords, avoiding social engineering attacks, and employee monitoring. *[Applying]*	Find policies that include procedures for managing passwords, avoiding social engineering attacks, and employee monitoring. *[Remembering]*	Write a security policy, which includes procedures for managing passwords, avoiding social engineering attacks, and employee monitoring. *[Applying]*	Compare policies, which include procedures for managing passwords, avoiding social engineering attacks, and employee monitoring. *[Analyzing]*

Bloom's Revised Taxonomy

The foundational Taxonomy of Educational Objectives: A Classification of Educational Goals was established in 1956 by Dr. Benjamin Bloom, an educational psychologist, and is often referred to as Bloom's Taxonomy. This classification divides educational objectives into three learning domains: Cognitive (knowledge), Affective (attitude) and Psychomotor (skills). In 2000, Lorin Anderson and David Krathwohl updated Bloom's seminal framework to create **Bloom's Revised Taxonomy**, focusing on the Cognitive and Affective Domains. As described below, the ACM Committee for Computing Education in Community Colleges (CCECC) has adopted **Bloom's Revised Taxonomy** for the assessment of student learning outcomes in its computing curricula.

Measurable Action Verbs of the Cognitive Domain

Remembering	***Understanding***	***Applying***	***Analyzing***	***Evaluating***	***Creating***
Define	*Classify*	*Apply*	*Analyze*	*Appraise*	*Assemble*
Duplicate	*Convert*	*Calculate*	*Attribute*	*Argue*	*Construct*
Find	*Demonstrate*	*Carry out*	*Categorize*	*Assess*	*Create*
Identify	*Describe*	*Edit*	*Compare*	*Choose*	*Design*
Label	*Differentiate*	*Diagram*	*Contrast*	*Critique*	*Develop*
List	*Discuss*	*Execute*	*Decompose*	*Debate*	*Devise*
Locate	*Exemplify*	*Illustrate*	*Deconstruct*	*Defend*	*Formulate*
Memorize	*Explain*	*Implement*	*Deduce*	*Estimate*	*Hypothesize*
Name	*Infer*	*Investigate*	*Discriminate*	*Evaluate*	*Invent*
Recall	*Interpret*	*Manipulate*	*Distinguish*	*Judge*	*Make*
Recognize	*Paraphrase*	*Modify*	*Examine*	*Justify*	*Plan*
Retrieve	*Report*	*Operate*	*Integrate*	*Support*	
Select	*Summarize*	*Perform*	*Organize*	*Test*	
State	*Translate*	*Produce*	*Outline*	*Value*	
		Solve	*Structure*	*Verify*	
		Use			
		Write			

Glossary of Terms

The ACM CCCECC defines the following terms in relationship to curricula associated with computing education in associate-degree granting institutions.

Associate Degrees are well-defined and meaningful completion points at the conclusion of two-year degree programs; such degrees are awarded by two-year, community or technical colleges, as well as some four-year colleges.

Career Programs are specifically designed to enable students to pursue entry into the workforce after two years of college studies; these are typically Associate of Applied Science (AAS) degree programs.

Computing is now recognized by the ACM as comprised of six defined sub-disciplines: computer science, computer engineering, software engineering, information systems, information technology, and cybersecurity.

Computer Engineering … involves the design and construction of processor-based systems comprised of hardware, software, and communications components. This four-year curriculum focuses on the synthesis of electrical engineering and computer science as applied to the design of systems such as cellular communications, consumer electronics, medical imaging and devices, alarm systems and military technologies. Upon graduation, students initiating careers as computer engineers should be able to design and implement systems that involve the integration of software and hardware devices.

Computer Science … involves design and innovation developed from computing principles. This four-year curriculum focuses on the theoretical foundations of computing, algorithms, and programming techniques, as applied to operating systems, artificial intelligence, informatics and the like. Upon graduation, students initiating careers as computer scientists should be prepared to work in a broad range of positions involving tasks from theoretical work to software development.

Cybersecurity … The ACM JTF defines cybersecurity as a “computing-based discipline involving technology, people, information, and processes to enable assured operations. It involves the creation, operation, analysis, and testing of secure computer systems. It is an interdisciplinary course of study, including aspects of law, policy, human factors, ethics, and risk management in the context of adversaries.” (*csec2017.org*)

Information Systems … involves the application of computing principles to business processes, bridging the technical and management fields. This four-year curriculum focuses on the design, implementation and testing of information systems as applied to business processes such as payroll, human resources, corporate databases, data warehousing and mining, e-commerce, finance, customer relations management, transaction processing, and data-driven decision making and executive support. Information systems graduates should be able to analyze information requirements and business processes and be able specify and design systems that are aligned with organizational goals.

Information Technology … involves the design, implementation and maintenance of technology solutions and support for users of such systems. This four-year curriculum focuses on crafting hardware and software solutions as applied to networks, security, client-server and mobile computing, web applications, multimedia resources, communications systems, and the planning and management of the technology lifecycle. Upon graduation, students initiating careers as information technology professionals should be able to work effectively at planning, implementation, configuration, and maintenance of an organization's computing infrastructure.

Software Engineering … involves the design, development and testing of large, complex, and safety-critical software applications. This four-year curriculum focuses on the integration of computer science principles with engineering practices as applied to constructing software systems for avionics, healthcare applications, cryptography, traffic control, meteorological systems and the like. Upon graduation, students initiating careers as software engineers should be able to properly perform and manage activities at every stage of the life cycle of large-scale software systems.

Transfer Programs are specifically designed for students intending to matriculate into the junior year of a four-year program; these are typically Associate of Arts (AA) or Associate of Science (AS) degree programs.

Bibliography

ACM Code of Ethics and Professional Conduct, (1992), Available from www.acm.org/about-acm/acm-code-of-ethics-and-professional-conduct

ACM/IEEE-CS Joint Curriculum Task Force, *Computer Science 2013: Curriculum Guidelines for Undergraduate Programs in Computer Science*, (2013), Available from www.cs2013.org.

ACM/IEEE-CS Joint Curriculum Task Force, Computing Curricula 2001: Computer Science, (2001).

ACM/IEEE-CS Joint Curriculum Task Force, Computing Curricula 1991, ACM Press and IEEE Computer Society Press (1991).

ACM Joint Task Force, Undergraduate Cybersecurity Curricular Guidelines (forthcoming 2017), Available from www.csec2017.org.

ACM Two-Year College Computing Curricula Task Force, Computing Curricula 2009: Guidelines for Associate-Degree Transfer Curriculum in Computer Science, ACM Press (2009).

ACM Two-Year College Computing Curricula Task Force, Computing Curricula 2003: Guidelines for Associate-Degree Curricula in Computer Science, ACM Press (2003).

ACM Two-Year College Computing Curricula Task Force, Computing Curricula Guidelines for Associate-Degree Programs: Computing Sciences. ACM Press (1993).

American Association of Community Colleges, http://www.aacc.nche.edu/

Anderson, L.W., and Krathwohl, D.A., *A Taxonomy for Learning, Teaching, and Assessing: A Revision to Bloom's Taxonomy of Educational Objectives*, (2001).

Bloom, Benjamin S., The Taxonomy of Educational Objectives: Classification of Educational Goals. Handbook I: The Cognitive Domain, McKay Press, New York (1956).

IEEE Computer Society, www.computer.org/education/

National Institute of Standards and Technology, National Initiative for Cybersecurity Educaiton (NICE), *The National Cybersecurity Workforce Framework*, (April 2014), Available from http://csrc.nist.gov/nice/framework/

The White House, Office of the Press Secretary, *FACT SHEET: National Cybersecurity Action Plan* (NCAP), (February 2016), Available from www.whitehouse.gov/the-press-office/2016/02/09/fact-sheet-cybersecurity-national-action-plan

Appendix A
Cybersecurity-related Learning Outcomes by Knowledge Area and Unit

AL/Algorithmic Strategies
- ❖ AL-07. Explain the role of using random/pseudo random number generation in cryptography.

AL/Fundamental Data Structures and Algorithms
- ❖ AL-16. Summarize the security vulnerabilities in various data structures.

AR/Machine Level Representation of Data
- ❖ AR-03. Explain how fixed-length number representations could affect accuracy, precision, and vulnerabilities.

GV/Fundamental Concepts
- ❖ GV-05. Illustrate the principle of information hiding through steganography in images, messages, videos, or other media files.

HCI/Foundations
- ❖ HCI-04. Investigate the issues of trust in HCI, including examples of both high and low trust systems.

HCI/Designing Interaction
- ❖ HCI-08. Illustrate the interaction between a security mechanism and its usability.

IAS/Foundational Concepts in Security
- ❖ IAS-01. Recognize the importance of security as a continuous process and its balancing nature between protection mechanisms and availability of data and information.
- ❖ IAS-02. Describe the concepts of risk, threats, vulnerabilities, attack vectors, and exploits (including the fact that there is no such thing as perfect security).
- ❖ IAS-03. Explain the importance of security controls and countermeasures to minimize security risk and exposure.
- ❖ IAS-04. Analyze the tradeoffs of balancing security properties (Confidentiality, Integrity, Availability, as well as Authentication, Authorization, Access, Authenticity, Non-Repudiation, Privacy).
- ❖ IAS-05. Explain the concepts of trust and trustworthiness.
- ❖ IAS-06. Describe important ethical issues to consider in security.
- ❖ IAS-07. Investigate risks to privacy and anonymity in technology.
- ❖ IAS-08. Apply cybersecurity principles to a changing landscape.
- ❖ IAS-09. Demonstrate the key role risk management frameworks play in identifying, assessing, prioritizing, and controlling risks that an organization’s assets face.

IAS/Principles of Secure Design

- IAS-10. Describe the principles of secure design.
- IAS-11. Discuss the implications of relying on open design or the secrecy of design for security.
- IAS-12. Explain the goals of end-to-end data security
- IAS-13. Discuss the benefits of having multiple layers of defenses (Defense in Depth).
- IAS-14. For each state in the lifecycle of a product, investigate what security consideration should be evaluated.
- IAS-15. Recognize the tradeoffs associated with designing security into a product.

IAS/Defensive Programming

- IAS-16. Implement input validation and data sanitization in applications as necessary considering adversarial control of the input channel.
- IAS-17. Explain the tradeoffs of developing a program in a type-safe language.
- IAS-18. Implement programs that properly handle exceptions and error conditions.
- IAS-19. Recognize the need to update software to fix security vulnerabilities.

IAS/Threats and Attacks

- IAS-20. Identify likely attack types against standalone and networked software systems.
- IAS-21. Describe risks to privacy and anonymity in information systems.
- IAS-22. Discuss the key principles, such as membership and trust, of social engineering.

IAS/Cryptography

- IAS-23. Explain the purpose of cryptography and how it is used to secure data.
- IAS-24. Define key terms in cryptology.
- IAS-25. Describe basic methods for transforming plaintext into ciphertext.
- IAS-26. Explain the difference between symmetric and asymmetric encryption and how they are collectively used to secure digital communications and e-commerce transactions.

IAS/Web Security

- IAS-27. Explain browser and web security model concepts including same-origin policy, web sessions, and secure communication channels.
- IAS-28. Investigate common vulnerabilities and attacks in web applications and the coding strategies that are used to mitigate them.

IAS/Secure Software Engineering

- IAS-29. Write software requirements that include basic security specifications.
- IAS-30. Implement a plan to test the security modules in software.
- IAS-31. Investigate security vulnerabilities in a software at the requirement and design phases of the software development lifecycle.

IM/Information Management Concepts

- IM-04. Describe types of contingency plans including business continuity, disaster recovery and incident response.
- IM-05. Describe proven techniques to secure data and information.

- IM-06. Investigate vulnerabilities and failure scenarios in information systems.

IM/Database Systems

- IM-11. Describe common security concerns in database management systems.
- IM-12. Apply security principles to the design and development of database systems.

IM/Data Modeling

- IM-18. Explain potential security and privacy concerns during the process of data modeling.

NC/Networked Applications

- NC-07. Describe security concerns in designing applications for use over wireless networks.
- NC-08. Discuss secure connectivity among networked applications.
- NC-09. Explain the advantages and disadvantages of using virtualized infrastructure in cloud computing.

OS/Security and Protection

- OS-11. Explain the need for protection and security in an Operating System (OS).
- OS-12. Discuss potential threats to operating systems and the security features designed to guard against them.

PD/Parallelism Fundamentals

- PD-03. Differentiate data races from higher level races.

PD/Communication and Coordination

- Explain mutual exclusion as a way to avoid a given race condition.

PL/Object-Oriented Programming

- PL-01. Implement a simple secure class hierarchy.
- PL-03. Use access modifiers to secure class data.
- PL-04. Describe the tenets of OOP, including encapsulation, abstraction, inheritance, and polymorphism and how they impact/influence security.

PL/Event-Driven and Reactive Programming

- PL-10. Describe potential security vulnerabilities in event-driven GUI applications.

PL/Basic Type Systems

- PL-12. Explain the security advantages of choosing a type-safe language for software development.

SDF/Algorithms and Design

- SDF-05. Explain the differences between brute-force and divide-and-conquer algorithms in relation to security objectives.

SDF/Fundamental Programming Concepts

- SDF-07. Investigate potential vulnerabilities in programming code.
- SDF-08. Write programs which use defensive programming techniques, including input validation, type checking, and protection against buffer overflow.

SDF/Development Methods

- SDF-18. Describe common coding errors that expose security vulnerabilities.
- SDF-25. Carry out a code review on a program component using a provided security checklist.

SE/Software Processes

- SE-03. Describe the phases of the secure software development lifecycle (SecSDLC).

SE/Software Project Management

- SE-05. Explain the risks in using third- party code.

SE/Software Design

- SE-12. Analyze an existing software design to improve its security.
- SE-13. Describe the cost and tradeoffs associated with designing security into software.
- SE-14. Explain first principles of software security, such as least privilege, information hiding, and simplicity.

SE/Software Verification and Validation

- SE-18. Describe security tests performed during software development.

SF/Computational Paradigms

- SF-05. Recognize security implications related to emerging computational paradigms.

SF/Cross-Layer Communications

- SF-08. Investigate defects in a layered program using tools for program tracing, single stepping, and debugging.

SP/Social Context

- SP-04. Describe potential social engineering attacks and the bad actors who might perform them.

SP/Professional Ethics

- SP-08. Investigate recent and historical legislation related to unlawful access and software piracy.

SP/Privacy and Civil Liberties

- SP-13. Investigate threats to privacy rights in personally identifiable information (PII).
- SP-15. Investigate technological solutions to privacy concerns.

SP/Security Policies, Laws and Computer Crime

- SP-21. List examples of computer crimes and social engineering incidents with societal impact.
- SP-22. Interpret laws that apply to computer crimes.
- SP-23. Describe the motivation and ramifications of cyber terrorism and criminal hacking.
- SP-24. Examine the ethical and legal issues surrounding the misuse of access and various breaches in security.
- SP-25. Write a company-wide policy, which includes procedures for managing passwords, avoiding social engineering attacks, and employee monitoring.

<** end of document **>