

PENNSYLVANIA STATEWIDE PROGRAM-TO-PROGRAM ARTICULATION AGREEMENT IN COMPUTER SCIENCE

In accordance with Act 50 of 2009, institutions participating in Pennsylvania's statewide college credit transfer system agree to the following policies governing the transfer of credits from a participating associate-degree granting institution into a participating four-year college or university. This agreement specifically ensures that a student who successfully completes an Associate of Arts (AA) or Associate of Science (AS) degree in Computer Science or any AA or AS degree that incorporates the required competencies at a participating institution can transfer the full degree into a parallel bachelor degree program in Computer Science at a participating four-year institution.

In order for students to transfer the full associate degree into a parallel bachelor degree program at a participating four-year institution, all of the following criteria must be met:

- Successful completion of an associate degree that includes all of the required major competencies identified in this Agreement.
- Successful completion of 30-credits of foundation courses from the Transfer Credit Framework.

See Appendix A: Program-to-Program Articulation Model for Computer Science.

It is therefore understood that students meeting these requirements will be considered by both the associate degree granting institution and the receiving four-year institution to possess the knowledge, skills and abilities necessary for entry as a junior into a parallel bachelor degree program in Computer Science.

REQUIRED Major-Specific Content Areas

Under this Agreement, a fully-transferable associate degree in the field of Computer Science must include competencies from three primary content areas:

- 1. Designing and Developing Programs**
- 2. Computer Organization**
- 3. Discrete Mathematics**
- 4. Calculus**
- 5. Statistics**

Institutions may determine how the competencies identified in these primary content areas are met. For example, one institution may choose an objects-first approach to teach programming while another may choose a functional language introduction to programming. How an institution incorporates the competencies into the associate degree program does not affect the transferability of the associate degree under this Agreement in so long as all of the competencies are met.

The competencies listed for the Computer Science and Discrete Mathematics content areas are patterned after the competencies given in "Computer Science Curriculum 2008: An Interim Revision of CS 2001"¹. The notation "{partial}" means that only a subset of the objectives for that area are required. The details of all of these requirements are included in the appendices.

1. Designing and Developing Programs

The following competencies have been identified as essential for comparable preparation in this content area:

1. PF/Fundamental programming constructs [core]
2. PF/Algorithms and problem-solving [core] {Partial coverage}
3. PF/Fundamental data structures [core] {partial }
4. PF/Recursion [core]
5. PF/Event Driven Programming [core]
6. PF/Object Oriented [core]
7. AL/Basic Analysis [core] {partial}
8. AL/Algorithm Strategies [core] {partial}
9. AL/Fundamental Algorithms [core] {partial}

¹ at <http://www.acm.org/education/curricula-recommendations>

10. PL/Overview [core]
11. PL/Virtual machines [core]
12. PL/Declarations and types [core]
13. PL/Object-oriented programming [core] {partial}
14. SP/History of computing [core]
15. SE/Software design [core] {partial}
16. SE2/Using APIs [core] {partial}
17. SE3/Software tools and environments [core] {partial}
18. SE5/Software requirements and specifications [core] {partial}
19. SE6/Software validation and verification [core] {partial}

See Appendix B: Competencies for Preparation in Designing and Developing Programs.

Historically, these competencies commonly have been found in courses such as Computer Science I, Computer Science II, and/or Data Structures. However, each associate institution may determine the format in which the competencies are presented.

2. Computer Organization

The following competencies have been identified as essential for comparable preparation in this content area:

1. AR/Digital Logic and Data Representation [core]
2. AR/Computer Architecture and Organization [core]
3. AR/Interfacing and I/O Strategies [core]
4. AR/Memory Architecture [core]
5. AR/Functional Organization [core]
6. AR/Multiprocessing [core]
7. AR/Performance Enhancements [elective]
8. AR/Distributed Architectures [elective]
9. AR/Devices [elective]
10. AR/Directions In Computing [elective]

See Appendix C. Competencies for Preparation in Computer Organization.

3. Discrete Mathematics

The following competencies have been identified as essential for comparable preparation in this content area:

1. DS/Functions, relations, and sets [core]
2. DS/Basic logic [core]
3. DS/Proof techniques [core]
4. DS/Basics of counting [core]
5. DS/Discrete probability [core]

See Appendix D: Competencies for Preparation in Discrete Mathematics.

4. Calculus

According to the Statewide Program-to-Program Articulation Agreement in Mathematics, the following competencies have been identified as essential for comparable preparation in this content area:

- | | |
|---------------|---|
| Competency 1: | Utilize the concept of limit. |
| Competency 2: | Differentiate functions. |
| Competency 3: | Use differential calculus to sketch curves and to solve applied problems. |
| Competency 4: | Integrate functions by approximation and by use of anti-derivatives. |
| Competency 5: | Use integral calculus to determine area and to solve applied problems. |

For detailed Competencies for Preparation in Calculus, see the Statewide Program-to-Program Articulation Agreement in Mathematics at www.PAcollegetransfer.com.

5. Statistics

The following competencies have been identified as essential for comparable preparation in this content area:

- | | |
|---------------|--|
| Competency 1: | Computing and interpreting statistical results |
| Competency 2: | Understand Probability Distributions |
| Competency 3: | Designing Experiments |
| Competency 4: | Read and Write Statistical Reports |

See Appendix F: Competencies for Preparation in Statistics.

REQUIRED Related-Content Area

In addition to the required competencies listed above, students transferring into advanced coursework in a bachelor degree program in Computer Science must also acquire competencies in at least ONE of the following content areas:

- 1. Operating Systems**
- 2. Databases**
- 3. Networks**

Competencies in these content areas are taken from “CC 2001: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.” The core areas must be mastered and a reasonable number of elective competencies must also be included.

1. Operating Systems

The following competencies have been identified as essential for comparable preparation in this content area:

1. OS/Overview Of Operating Systems [core]
2. OS/Operating System Principles [core]
3. OS/Concurrency [core]
4. OS/Scheduling and Dispatch [core]
5. OS/Memory Management [core]
6. OS/Device Management [elective]
7. OS/Security And Protection [core]
8. OS/File Systems [elective]
9. OS/Real Time And Embedded Systems [elective]
10. OS/Fault Tolerance [elective]
11. OS/System Performance Evaluation [elective]
12. OS/Scripting [elective]
13. OS/Digital Forensics [elective]
14. OS/Security Models [elective]

See Appendix G: Competencies in Operating Systems.

2. Databases

The following competencies have been identified as essential for comparable preparation in this content area:

1. IM/Information Models [core]
2. IM/Database Systems [core]
3. IM/Data Modeling [core]

See Appendix H: Competencies in Databases.

3. Networks

The following competencies have been identified as essential for comparable preparation in this content area:

1. Net Centric Computing
2. NC/Network Communication [core]
3. NC/Network Security [core]
4. NC/Web Organization [Elective]
5. NC/Networked Applications [elective]
6. NC/Network Management [elective]
7. NC/Compression [Elective]
8. NC/Multimedia Technologies] [elective]
9. NC/Mobile Computing [elective]

See Appendix I: Competencies in Networks.

REQUIRED General Coursework

Associate degree students transferring into a bachelor degree program in Computer Science must also complete coursework outside of the major. Specifically, students are required to take a two-semester science sequence targeted to science majors in order to facilitate transfer into ABET-accredited programs

It is recommended that the students take a science sequence matching the Accreditation Board for Engineering and Technology (ABET) requirements for such a sequence: "A science component that develops an understanding of the scientific method and provides students with an opportunity to experience this mode of inquiry in courses for science or engineering majors that provide some exposure to laboratory work."

See Appendix A: Program-to-Program Articulation Model for Computer Science.

Transfer Credit Framework

Under Act 114 of 2006, the Commonwealth's statewide college credit transfer system includes an advising tool called the "Transfer Credit Framework". The Framework allows students to seamlessly transfer up to 30 credits of foundation courses from one participating college or university to another and have those courses count toward graduation. See Appendix J Transfer Credit Framework.

The Framework consists of a menu of 49 courses that fall within six broad categories: English, public speaking, math, science, fine arts and humanities, and the behavioral and social sciences. To fully benefit from the Framework, students are advised to select a range of courses according to the criteria for each category.

Under this Agreement, students may select courses according to the criteria indicated for Framework Category 1, Category 2, Category 5 and Category 6.

In Framework Category 3, students may apply a maximum of 4 credits (1 course) completed as part of the Required Major-Specific Content Area of Discrete Mathematics and a maximum of 4 credits (1 course) completed as part of the Required Major-Specific Content Area Related Mathematics.

Likewise, a maximum of 8 credits (2 courses) in a lab-based science sequence, recommended coursework outside of the field of Mathematics, may be used to satisfy the requirements of Framework Category 4. See Appendix A: Program-to-Program Articulation Model for Computer Science.

Students are advised to work with an advisor to select courses related to their associate degree program, transfer major and personal interests.

Appendix A: Program-to-Program Articulation Model for Computer Science

REQUIRED Major-Specific Content Areas	Transfer Criteria
Designing and Developing Programs	Completion of courses addressing the required competencies specified in this Agreement for Design and Developing Programs.
Computer Organization	Completion of courses addressing the required competencies specified in this Agreement for Computer Organization.
Discrete Mathematics	<ul style="list-style-type: none"> • Completion of courses addressing the required competencies specified in this Agreement for Discrete Mathematics. • Computer Science majors may use credits from this requirement to satisfy the requirements of Category 3 in the Transfer Credit Framework.
Calculus	<ul style="list-style-type: none"> • Completion of courses addressing the required competencies specified in this Agreement for Calculus. • Computer Science majors may use credits from this requirement to satisfy the requirements of Category 3 in the Transfer Credit Framework.
Statistics	Completion of courses addressing the required competencies specified in this Agreement for Statistics.
REQUIRED Related Content Areas	Transfer Criteria
Students must complete coursework <u>in at least ONE</u> of the following content areas: <ol style="list-style-type: none"> 1. Operating Systems 2. Databases 3. Networks 	Completion courses addressing the required competencies specified in this Agreement for each area.
REQUIRED Coursework Outside of the Discipline	Transfer Criteria
2 courses in a Lab-based Science Sequence for Science Majors	<ul style="list-style-type: none"> • Computer Science majors may use this coursework to satisfy the requirements of Category 4 of the Transfer Credit Framework. See below. • Student should consult an advisor before enrolling in the coursework.
TRANSFER CREDIT FRAMEWORK	Transfer Criteria
Category 1	1 course to be selected by the student with the assistance of an advisor
Category 2	1 course to be selected by the student with the assistance of an advisor
Category 3	<ol style="list-style-type: none"> 1. Calculus I 2. Discrete Mathematics
Category 4	2 courses of a Lab-based Science Sequence for Science Majors
Category 5	2 courses to be selected by the student with the assistance of an advisor
Category 6	2 courses to be selected by the student with the assistance of an advisor

Appendix B: Competencies for Preparation in Designing and Developing Programs

PF/Fundamental programming constructs [core]

Topics:

1. Basic syntax and semantics of a higher-level language
2. Variables, types, expressions, and assignment
3. Simple I/O
4. Conditional and iterative control structures
5. Functions and parameter passing
6. Structured decomposition

Competencies:

1. Analyze and explain the behavior of simple programs involving the fundamental programming constructs covered by this unit.
2. Modify and expand short programs that use standard conditional and iterative control structures and functions.
3. Design, implement, test, and debug a program that uses each of the following fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, and the definition of functions.
4. Choose appropriate conditional and iteration constructs for a given programming task.
5. Apply the techniques of structured (functional) decomposition to break a program into smaller pieces.
6. Describe the mechanics of parameter passing.

PF/Algorithms and problem-solving [core] {Partial coverage}

Topics:

1. Problem-solving strategies
2. The role of algorithms in the problem-solving process
3. Implementation strategies for algorithms
4. Debugging strategies
5. The concept and properties of algorithms

Competencies:

1. Discuss the importance of algorithms in the problem-solving process.
2. Identify the necessary properties of good algorithms.
3. Create algorithms for solving simple problems.
4. Use pseudocode or a programming language to implement, test, and debug algorithms for solving simple problems.
5. Describe strategies that are useful in debugging.

PF3/Fundamental data structures [core] {partial}

Topics:

1. Primitive types
2. Arrays
3. Records
4. Strings and string processing
5. Data representation in memory
6. Static, stack, and heap allocation
7. Runtime storage management
8. Pointers and references
9. Linked structures
10. Implementation strategies for stacks, queues, and hash tables
11. Implementation strategies for graphs and trees
12. Strategies for choosing the right data structure

Competencies:

1. Describe the representation of numeric and character data.
2. Understand how precision and round-off can affect numeric calculations.
3. Discuss the use of primitive data types and built-in data structures.
4. Describe common applications for each data structure in the topic list.
5. Implement the user-defined data structures in a high-level language.
6. Compare alternative implementations of data structures with respect to performance.
7. Write programs that use each of the following data structures: arrays, strings, linked lists, stacks, queues, and hash tables.
8. Compare and contrast the costs and benefits of dynamic and static data structure implementations.
9. Choose the appropriate data structure for modeling a given problem.

PF/ Recursion [core]

Topics:

1. The concept of recursion
2. Recursive mathematical functions
3. Simple recursive procedures
4. Divide-and-conquer strategies
5. Recursive backtracking

Competencies:

1. Describe the concept of recursion and give examples of its use.
2. Identify the base case and the general case of a recursively defined problem.
3. Compare iterative and recursive solutions for elementary problems such as factorial.
4. Describe the divide-and-conquer approach.
5. Implement, test, and debug simple recursive functions and procedures.
6. Determine when a recursive solution is appropriate for a problem.

PF/Event Driven Programming [core]

Topics:

1. Event-handling methods
2. Event propagation
3. Exception handling

Competencies:

1. Explain the difference between event-driven programming and command-line programming.
2. Design, code, test, and debug simple event-driven programs that respond to user events.
3. Develop code that responds to exception conditions raised during execution.

PF/Object Oriented [core]

Topics:

1. Object-oriented design
2. Encapsulation and information-hiding
3. Separation of behavior and implementation
4. Classes and subclasses
5. Inheritance (overriding, dynamic dispatch)
6. Polymorphism (subtype polymorphism vs. inheritance)

Competencies:

1. Justify the philosophy of object-oriented design and the concepts of encapsulation, abstraction, inheritance, and polymorphism.
2. Design, implement, test, and debug simple programs in an object-oriented programming language.
3. Describe how the class mechanism supports encapsulation and information hiding.
4. Design, implement, and test the implementation of “is-a” relationships among objects using a class hierarchy and inheritance.
5. Compare and contrast the notions of overloading and overriding methods in an object-oriented language.

AL/Basic Analysis [core] {partial}

Topics

1. Asymptotic analysis of upper and average complexity bounds
2. Identifying differences among best, average, and worst case behaviors
3. Standard complexity classes
4. Empirical measurements of performance
5. Time and space tradeoffs in algorithms

Competencies:

1. Explain the use of big O to describe the amount of work done by an algorithm.
2. Use big O notation to give asymptotic upper bounds on time and space complexity of algorithms.
3. Determine the time and space complexity of simple algorithms.

AL/Fundamental Algorithms [core] {partial}

Topics:

1. Simple numerical algorithms
2. Sequential and binary search algorithms
3. Quadratic sorting algorithms (selection, insertion)
4. $O(N \log N)$ sorting algorithms (Quicksort, heapsort, mergesort)
5. Hash tables, including collision-avoidance strategies
6. Binary search trees

Competencies:

1. Implement the most common quadratic and $O(N \log N)$ sorting algorithms.
2. Design and implement an appropriate hashing function for an application.
3. Design and implement a collision-resolution algorithm for a hash table.
4. Discuss the computational efficiency of the principal algorithms for sorting, searching, and hashing.
5. Discuss factors other than computational efficiency that influence the choice of algorithms, such as programming time, maintainability, and the use of application-specific patterns in the input data.

PL/Overview [core]

Topics:

1. History of programming languages
2. Brief survey of programming paradigms
 - i. Procedural languages
 - ii. Object-oriented languages
 - iii. Functional languages
 - iv. Declarative, non-algorithmic languages
 - v. Scripting languages
2. The effects of scale on programming methodology

Competencies:

1. Summarize the evolution of programming languages illustrating how this history has led to the paradigms available today.
2. Identify at least one distinguishing characteristic for each of the programming paradigms covered in this unit.
3. Evaluate the tradeoffs between the different paradigms, considering such issues as space efficiency, time efficiency (of both the computer and the programmer), safety, and power of expression.
4. Distinguish between programming-in-the-small and programming-in-the-large.

PL/Virtual machines [core]

Topics:

1. The concept of a virtual machine
2. Hierarchy of virtual machines
3. Intermediate languages
4. Security issues arising from running code on an alien machine

Competencies:

1. Describe the importance and power of abstraction in the context of virtual machines.
2. Explain the benefits of intermediate languages in the compilation process.
3. Evaluate the tradeoffs in performance vs. portability.

4. Explain how executable programs can breach computer system security by accessing disk files and memory.

PL/Declarations and types [core]

Topics:

1. The conception of types as a set of values with together with a set of operations
2. Declaration models (binding, visibility, scope, and lifetime)
3. Overview of type-checking
4. Garbage collection

Competencies:

1. Explain the value of declaration models, especially with respect to programming-in- the-large.
2. Identify and describe the properties of a variable such as its associated address, value, scope, persistence, and size.
3. Discuss type incompatibility.
4. Demonstrate different forms of binding, visibility, scoping, and lifetime management.CC2001 Computer Science volume – 115 – Final Report (December 15, 2001)
5. Defend the importance of types and type-checking in providing abstraction and safety.
6. Evaluate tradeoffs in lifetime management (reference counting vs. garbage collection).

PL/Object-oriented programming [core] {partial}

Topics:

1. Object-oriented design
2. Encapsulation and information-hiding
3. Separation of behavior and implementation
4. Classes and subclasses
5. Inheritance (overriding, dynamic dispatch)
6. Polymorphism (subtype polymorphism vs. inheritance)
7. Class hierarchies
8. Collection classes and iteration protocols
9. Internal representations of objects and method tables

Competencies:

1. Justify the philosophy of object-oriented design and the concepts of encapsulation, abstraction, inheritance, and polymorphism.
2. Design, implement, test, and debug simple programs in an object-oriented programming language.
3. Describe how the class mechanism supports encapsulation and information hiding.
4. Design, implement, and test the implementation of “is-a” relationships among objects using a class hierarchy and inheritance.
5. Compare and contrast the notions of overloading and overriding methods in an object-oriented language.
6. Explain the relationship between the static structure of the class and the dynamic structure of the instances of the class.
7. Describe how iterators access the elements of a container.

SP/History of computing [core]

Topics:

1. Prehistory—the world before 1946
2. History of computer hardware, software, networking
3. Pioneers of computing

Competencies:

1. List the contributions of several pioneers in the computing field.
2. Compare daily life before and after the advent of personal computers and the Internet.
3. Identify significant continuing trends in the history of the computing field.

SE/Software design [core] {partial}

Topics:

1. Fundamental design concepts and principles
2. Object-oriented analysis and design
3. Design for reuse

Competencies:

1. Discuss the properties of good software design.
2. Evaluate the quality of multiple software designs based on key design principles and concepts.
3. Create and specify the software design for a medium-size software product using a software requirement specification, an accepted program design methodology (e.g., structured or object-oriented), and appropriate design notation.
4. Evaluate a software design at the component level.
5. Evaluate a software design from the perspective of reuse.

SE/Using APIs [core] {partial}

Topics:

1. Programming Using APIs
2. Class browsers and related tools
3. Debugging in the API environment

Competencies:

1. Explain the value of application programming interfaces (APIs) in software development.
2. Use class browsers and related tools during the development of applications using APIs.
3. Design, implement, test, and debug programs that use large-scale API packages.

SE/Tools and environments [core] {partial}

Topics:

1. Programming environments
2. Testing tools

Competencies:

1. Select, with justification, an appropriate set of tools to support the development of a range of software products.
2. Demonstrate the capability to use a range of software tools in support of the development of a software product of medium size.

SE/Software requirements and specifications [core] {partial}

Topics:

1. The importance of specification in the software process

Competencies:

1. Use a common, non-formal method to model and specify (in the form of a requirements specification document) the requirements for a medium-size software system.

SE/Software verification and validation [core] {partial}

Topics:

1. Testing fundamentals, including test case generation

Competencies:

1. Describe the role that tools can play in the validation of software.
2. Discuss the issues involving the testing of object-oriented software.

Appendix C: Competencies for Preparation in Computer Organization

The computer lies at the heart of computing. Without it most of the computing disciplines today would be a branch of theoretical mathematics. A professional in any field of computing should not regard the computer as just a black box that executes programs by magic. All students of computing should acquire some understanding and appreciation of a computer system's functional components, their characteristics, their performance, and their interactions. Students need to understand computer architecture in order to make best use of the software tools and computer languages they use to create programs. In this introduction the term architecture is taken to include instruction set architecture (the programmer's abstraction of a computer), organization or microarchitecture (the internal implementation of a computer at the register and functional unit level), and system architecture (the organization of the computer at the cache, and bus level). Students should also understand the complex tradeoffs between CPU clock speed, cache size, bus organization, number of core processors, and so on. Computer architecture also underpins other areas of the computing curriculum such as operating systems (input/output, memory technology) and high-level languages (pointers, parameter passing).

The learning objectives specified for these topics correspond primarily to the core and are intended to support programs that require only the minimum 36 hours of computer architecture. For programs that want to teach more than the minimum, the same topics (AR1-AR6) can be treated at a more advanced level by implementing a two course sequence. For programs that want to cover the elective topics, those topics can be introduced within a two course sequence and/or be treated in a more comprehensive way in a third course.

- **AR. Architecture and Organization (36 core hours)**
- **AR/Digital Logic and Data Representation [core]**
- **AR/ Computer Architecture and Organization [core]**
- **AR/ Interfacing and I/O Strategies [core]**
- **AR/Memory Architecture [core]**
- **AR/Functional Organization [core]**
- **AR/Multiprocessing [core]**
- **AR/Performance Enhancements [elective]**
- **AR/Distributed Architectures [elective]**
- **AR/Devices [elective]**
- **AR/Directions In Computing [elective]**

AR/Digital Logic and Data Representation [core]

Topics:

1. Introduction to digital logic (logic gates, flip-flops, circuits)
2. Logic expressions and Boolean functions
3. Representation of numeric data
4. Signed and unsigned arithmetic
5. Range, precision, and errors in floating-point arithmetic
6. Representation of text, audio, and images
7. Data compression

Competencies:

1. Design a simple circuit using fundamental building blocks.
2. Appreciate the effect of AND, OR, NOT and EOR operations on binary data
3. Understand how numbers, text, images, and sound can be represented in digital form and the limitations of such representations
4. Understand how errors due to rounding effects and their propagation affect the accuracy of chained calculations.
5. Appreciate how data can be compressed to reduce storage requirements including the concepts of lossless and lossy compression.

AR/Computer Architecture and Organization [core]

Topics:

1. Overview of the history of the digital computer
2. Introduction to instruction set architecture, micro architecture and system architecture
3. Processor architecture – instruction types, register sets, addressing modes
4. Processor structures – memory-to-register and load/store architectures
5. Instruction sequencing, flow-of-control, subroutine call and return mechanisms

6. Structure of machine-level programs
7. Limitations of low-level architectures
8. Low-level architectural support for high-level languages

Competencies:

1. Describe the progression of computers from vacuum tubes to VLSI.
2. Appreciate the concept of an instruction set architecture, ISA, and the nature of a machine-level instruction in terms of its functionality and use of resources (registers and memory).
3. To understand the relationship between instruction set architecture, micro architecture, and system architecture and their roles in the development of the computer.
4. Be aware of the various classes of instruction: data movement, arithmetic, logical, and flow control.
5. Appreciate the difference between register-to-memory ISAs and load/store ISAs.
6. Appreciate how conditional operations are implemented at the machine level.
7. Understand the way in which subroutines are called and returns made.
8. Appreciate how a lack of resources in ISPs has an impact on high-level languages and the design of compilers.
9. Understand how, at the assembly language level, how parameters are passed to subroutines and how local workspace is created and accessed.

AR/Interfacing and I/O Strategies [core]

Topics:

1. I/O fundamentals: handshaking and buffering
2. Interrupt mechanisms: vectored and prioritized, interrupt acknowledgment
3. Buses: protocols, arbitration, direct-memory access (DMA)
4. Examples of modern buses: e.g., PCIe, USB, Hypertransport

Competencies:

1. Appreciate the need of open- and closed-loop communications and the use of buffers to control dataflow.
2. Explain how interrupts are used to implement I/O control and data transfers.
3. Identify various types of buses in a computer system and understand how devices compete for a bus and are granted access to the bus.
4. Be aware of the progress in bus technology and understand the features and performance of a range of modern buses (both serial and parallel).

AR/Memory Architecture [core]

Topics:

1. Storage systems and their technology (semiconductor, magnetic)
2. Storage standards (CD-ROM, DVD)
3. Memory hierarchy, latency and throughput
4. Cache memories - operating principles, replacement policies, multilevel cache, cache coherency

Competencies:

1. Identify the memory technologies found in a computer and be aware of the way in which memory technology is changing.
2. Appreciate the need for storage standards for complex data storage mechanisms such as DVD.
3. Understand why a memory hierarchy is necessary to reduce the effective memory latency.
4. Appreciate that most data on the memory bus is cache refill traffic
5. Describe the various ways of organizing cache memory and appreciate the cost-performance tradeoffs for each arrangement.
6. Appreciate the need for cache coherency in multiprocessor systems

AR/Functional Organization [core]

Topics:

1. Review of register transfer language to describe internal operations in a computer
2. Microarchitectures - hardwired and microprogrammed realizations
3. Instruction pipelining and instruction-level parallelism (ILP)
4. Overview of superscalar architectures
5. Processor and system performance
6. Performance – their measures and their limitations
7. The significance of power dissipation and its effects on computing structures

Competencies:

1. Review of the use of register transfer language to describe internal operations in a computer
2. Understand how a CPU's control unit interprets a machine-level instruction – either directly or as a microprogram.
3. Appreciate how processor performance can be improved by overlapping the execution of instruction by pipelining.
4. Understand the difference between processor performance and system performance (i.e., the effects of memory systems, buses and software on overall performance).
5. Appreciate how superscalar architectures use multiple arithmetic units to execute more than one instruction per clock cycle.
6. Understand how computer performance is measured by measurements such as MIPS or SPECmarks and the limitations of such measurements.
7. Appreciate the relationship between power dissipation and computer performance and the need to minimize power consumption in mobile applications.

AR/Multiprocessing [core]

Topics:

1. Amdahl's law
2. Short vector processing (multimedia operations)
3. Multicore and multithreaded processors
4. Flynn's taxonomy: Multiprocessor structures and architectures
5. Programming multiprocessor systems
6. GPU and special-purpose graphics processors
7. Introduction to reconfigurable logic and special-purpose processors

Competencies:

1. Discuss the concept of parallel processing and the relationship between parallelism and performance.
2. Appreciate that multimedia values (e.g., 8-/16-bit audio and visual data) can be operated on in parallel in 64-bit registers to enhance performance.
3. Understand how performance can be increased by incorporating multiple processors on a single chip.
4. Appreciate the need to express algorithms in a form suitable for execution on parallel processors.
5. Understand how special-purpose graphics processors, GPUs, can accelerate performance in graphics applications.
6. Understand the organization of computer structures that can be electronically configured and reconfigured

AR/Performance Enhancements [elective]

Topics:

1. Branch prediction
2. Speculative execution
3. Superscalar architecture
4. Out-of-order execution
5. Multithreading
6. Scalability
7. Introduction to VLIW and EPIC architectures
8. Memory access ordering

Competencies:

1. Explain the concept of branch prediction its use in enhancing the performance of pipelined machines.
2. Understand how speculative execution can improve performance.
3. Provide a detailed description of superscalar architectures and the need to ensure program correctness when executing instructions out-of-order.
4. Explain speculative execution and identify the conditions that justify it.
5. Discuss the performance advantages that multithreading can offer along with the factors that make it difficult to derive maximum benefits from this approach.
6. Appreciate the nature of VLIW and EPIC architectures and the difference between them (and between superscalar processors)
7. Understand how a processor re-orders memory loads and stores to increase performance.

AR/Distributed Architectures [elective]

Topics:

1. Introduction to LANs and WANs and the history of networking and the Internet
2. Layered protocol design, network standards and standardization bodies
3. Network computing and distributed multimedia
4. Mobile and wireless computing
5. Streams and datagrams
6. Physical layer networking concepts
7. Data link layer concepts (framing, error control, flow control, protocols)
8. Internetworking and routing (routing algorithms, internetworking, congestion control)
9. Transport layer services (connection establishment, performance issues)

Competencies:

1. Explain the basic components of network systems and distinguish between LANs and WANs.
2. Discuss the architectural issues involved in the design of a layered network protocol.
3. Explain how architectures differ in network and distributed systems.
4. Appreciate the special requirements of wireless computing.
5. Understand the difference between the roles of the physical layer and data link layer and appreciate how imperfections in the physical layer are handled by the data link layer.
6. Describe emerging technologies in the net-centric computing area and assess their current capabilities, limitations, and near-term potential.
7. Understand how the network layer can detect and correct errors.

AR/Devices [elective]

Topics:

1. Representing analog values digitally – quantization and sampling
2. Sound and audio, image and graphics, animation and video
3. Multimedia standards (audio, music, graphics, image, telephony, video, TV)
4. Input transducers (temperature, pressure, position, movement)
5. Input devices: mice, keyboards (text and musical), scanners, touch-screen, voice
6. Output devices: displays, printers
7. Encoding and decoding of multimedia systems including compression and decompression
8. Example of computer-based systems: GPS, MP3 players, digital cameras

Competencies:

1. Understand how analog quantities such as pressure can be represented in digital form and how the use of a finite representation leads to quantization errors.
2. Appreciate the need for multimedia standards and be able to explain in non-technical language what the standard calls for.
3. Understand how multimedia signals usually have to be compressed to conserve bandwidths using lossless or lossy encoding.
4. Discuss the design, construction, and operating principles of transducers such as Hall-effect devices and strain gauges
5. Appreciate how typical input devices operate.
6. Understand the principles of operation and performance of various display devices.
7. Study the operation of high-performance computer-based devices such as digital cameras

AR/Directions in Computing [elective]

Topics:

1. Semiconductor technology and Moore's law
2. Limitations to semiconductor technology
3. Quantum computing
4. Optical computing
5. Molecular (biological) computing
6. New memory technologies

Competencies:

1. To appreciate the underlying physical basis of modern computing.
2. Understand how the physical properties of matter impose limitations on computer technology
3. Appreciate how the quantum nature of matter can be exploited to permit massive parallelism

4. Appreciate how light can be used to perform certain types of computation
5. Understand how the properties of complex molecules can be exploited by organic computers
6. To get an insight into trends in memory design such as ovonic memory and ferromagnetic memories

Appendix D: Competencies for Preparation in Discrete Mathematics

DS/Functions, relations, and sets [core]

Topics:

1. Functions (surjections, injections, inverses, composition)
2. Relations (reflexivity, symmetry, transitivity, equivalence relations)
3. Sets (Venn diagrams, complements, Cartesian products, power sets)
4. Pigeonhole principle
5. Cardinality and countability

Competencies:

1. Explain with examples the basic terminology of functions, relations, and sets.
2. Perform the operations associated with sets, functions, and relations.
3. Relate practical examples to the appropriate set, function, or relation model, and interpret the associated operations and terminology in context.
4. Demonstrate basic counting principles, including uses of diagonalization and the pigeonhole principle.

DS/Basic logic [core]

Topics:

1. Propositional logic
 - a. Logical connectives
 - b. Truth tables
 - c. Normal forms (conjunctive and disjunctive)
 - d. Validity
2. Predicate logic
 - a. Universal and existential quantification
 - b. Modus ponens and modus tollens
 - c. Limitations of predicate logic

Competencies:

1. Apply formal methods of symbolic propositional and predicate logic.
2. Describe how formal tools of symbolic logic are used to model algorithms and real- life situations.
3. Use formal logic proofs and logical reasoning to solve problems such as puzzles. 4. Describe the importance and limitations of predicate logic.

DS/Proof techniques [core]

Topics:

1. The structure of formal proofs
2. Direct proofs
3. Proof by counterexample
4. Proof by contradiction
5. Mathematical induction
6. Strong Induction
7. Recursive mathematical definitions
8. Well orderings

Competencies:

1. Outline the basic structure of and give examples of each proof technique described in this unit.
2. Discuss which type of proof is best for a given problem.
3. Relate the ideas of mathematical induction to recursion and recursively defined structures.
4. Identify the difference between mathematical and strong induction and give examples of the appropriate use of each.

DS/Basics of counting [core]

Topics:

1. Counting arguments
2. Sum and product rule
3. Inclusion-exclusion
4. Arithmetic and geometric progressions
5. Fibonacci numbers
6. The pigeonhole principle
7. Permutations and combinations
 - a. Basic definitions
 - b. Pascal's identity
 - c. The binomial theorem
8. Solving recurrence relations
 - a. Common examples
 - b. The Master theorem

Competencies:

1. Compute permutations and combinations of a set, and interpret the meaning in the context of the particular application.
2. State the definition of the Master theorem.
3. Solve a variety of basic recurrence equations.
4. Analyze a problem to create relevant recurrence equations or to identify important counting questions.

DS/GraphsAndTrees [core]

Topics:

1. Trees
2. Undirected graphs
3. Directed graphs
4. Spanning trees/forests
5. Traversal strategies

Competencies:

1. Illustrate by example the basic terminology of graph theory, and some of the properties and special cases of each.
2. Demonstrate different traversal methods for trees and graphs.
3. Model problems in computer science using graphs and trees.
4. Relate graphs and trees to data structures, algorithms, and counting.

DS/Discrete probability [core]

Topics:

1. Finite probability space, probability measure, events
2. Conditional probability, independence, Bayes' theorem
3. Integer random variables, expectation
4. Law of large numbers

Competencies:

1. Calculate probabilities of events and expectations of random variables for elementary problems such as games of chance.
2. Differentiate between dependent and independent events.
3. Apply the binomial theorem to independent events and Bayes theorem to dependent events.
4. Apply the tools of probability to solve problems such as the Monte Carlo method, the average case analysis of algorithms, and hashing.

Appendix E: Competencies for Preparation in Calculus

The details of these competencies can be found in the Statewide Program-to-Program Articulation Agreement in Mathematics approved November 2010.

Appendix F: Competencies for Preparation in Statistics

Competency 1: Computing and interpreting statistical results

Topics:

1. Descriptive Statistics
 - a. Measures of Center, Position and Spread
 - b. Normal Distributions
 - c. Scatterplots and Regression
 - d. Data Analysis for Two-Way Tables

Competencies:

1. Use a statistical software package for the computer to compute statistical results
2. Interpret the values of descriptive statistics and what they say about a population

Competency 2: Understand Probability Distributions

Topics:

1. Probability
 - a. Basic Probability
 - b. Probability Rules
 - c. Conditional Probabilities and Bayes' Rule
2. Random Variables
 - a. Distributions
 - b. Means
 - c. Variance
3. Sampling Distributions
 - a. Counts
 - b. Proportions
 - c. Means

Competencies:

1. Use probability rules to combine probabilities
2. Apply Bayes rule
3. Define various random distributions, know what they model, and how to calculate descriptive statistics for them.
4. Understand the concept of sampling and ways it can be approached

Competency 3: Designing Experiments

Topics:

1. Sampling and Experimental Design
2. Confidence Intervals
 - a. Means
 - b. Proportions
 - c. Two Means
 - d. Two Proportions
3. Hypothesis Tests
 - a. Means
 - b. Proportions
 - c. Two Means
 - d. Two Proportions
 - e. Matched Pairs
4. Paired t-test
5. Sign Test for Difference
 - a. Chi-Square
 - b. Chi-Square Goodness of Fit
 - c. ANOVA (one-way)
 - d. Nonparametric
6. Wilcoxon Rank Sum
7. Wilcoxon Signed Rank
8. Kruskal-Wallis

Competencies:

1. Collect data from a real life population
2. Organize data to support statistical analysis
3. Compute and interpret confidence intervals
4. Set up and execute appropriate hypothesis tests
3. Make inferences from data to draw conclusions about populations

Competency 4: Read and Write Statistical Reports

Competencies:

1. Read and analyze statistical reports
2. Prepare statistical reports from experimental results

Appendix G: Competencies for Preparation in Operating Systems

OS / Overview of Operating Systems [core]

Topics:

1. Role and purpose of the operating system
2. History of operating system development
3. Functionality of a typical operating system
4. Mechanisms to support client-server models, hand-held devices
5. Design issues (efficiency, robustness, flexibility, portability, security, compatibility)
6. Influences of security, networking, multimedia, windows

Competencies:

1. Explain the objectives and functions of modern operating systems.
2. Describe how operating systems have evolved over time from primitive batch systems to sophisticated multiuser systems.
3. Analyze the tradeoffs inherent in operating system design.
4. Describe the functions of a contemporary operating system with respect to convenience, efficiency, and the ability to evolve.
5. Discuss networked, client-server, distributed operating systems and how they differ from single user operating systems.
6. Identify potential threats to operating systems and the security features design to guard against them.
7. Describe how issues such as open source software and the increased use of the Internet are influencing operating system design.

OS / Operating System Principles [core]

Topics:

1. Structuring methods (monolithic, layered, modular, micro-kernel models)
2. Abstractions, processes, and resources
3. Concepts of application program interfaces (APIs)
4. Application needs and the evolution of hardware/software techniques
5. Device organization • Interrupts: methods and implementations
6. Concept of user/system state and protection, transition to kernel mode

Competencies:

1. Explain the concept of a logical layer.
2. Explain the benefits of building abstract layers in hierarchical fashion.
3. Defend the need for APIs and middleware.
4. Describe how computing resources are used by application software and managed by system software.
5. Contrast kernel and user mode in an operating system.
6. Discuss the advantages and disadvantages of using interrupt processing.
7. Compare and contrast the various ways of structuring an operating system such as object-oriented, modular, micro-kernel, and layered.
8. Explain the use of a device list and driver I/O queue.

OS / Concurrency [core]

Topics:

1. States and state diagrams
2. Structures (ready list, process control blocks, and so forth)
3. Dispatching and context switching
4. The role of interrupts
5. Concurrent execution: advantages and disadvantages
6. The "mutual exclusion" problem and some solutions
7. Deadlock: causes, conditions, prevention
8. Models and mechanisms (semaphores, monitors, condition variables, rendezvous)
9. Producer-consumer problems and synchronization
10. Multiprocessor issues (spin-locks, reentrancy)

Competencies:

1. Describe the need for concurrency within the framework of an operating system.
2. Demonstrate the potential run-time problems arising from the concurrent operation of many separate tasks.
3. Summarize the range of mechanisms that can be employed at the operating system level to realize concurrent systems and describe the benefits of each.
4. Explain the different states that a task may pass through and the data structures needed to support the management of many tasks.
5. Summarize the various approaches to solving the problem of mutual exclusion in an operating system.
6. Describe reasons for using interrupts, dispatching, and context switching to support concurrency in an operating system.
7. Create state and transition diagrams for simple problem domains.
8. Discuss the utility of data structures, such as stacks and queues, in managing concurrency.
9. Explain conditions that lead to deadlock.

OS / Scheduling And Dispatch [core]

Topics:

1. Preemptive and non-preemptive scheduling
2. Schedulers and policies
3. Processes and threads
4. Deadlines and real-time issues

Competencies:

1. Compare and contrast the common algorithms used for both preemptive and non-preemptive scheduling of tasks in operating systems, such as priority, performance comparison, and fair-share schemes.
2. Describe relationships between scheduling algorithms and application domains.
3. Discuss the types of processor scheduling such as short-term, medium-term, long-term, and I/O.
4. Describe the difference between processes and threads.
5. Compare and contrast static and dynamic approaches to real-time scheduling.
6. Discuss the need for preemption and deadline scheduling.
7. Identify ways that the logic embodied in scheduling algorithms are applicable to other domains, such as disk I/O, network scheduling, project scheduling, and other problems unrelated to computing.

OS / Memory Management [core]

Topics:

1. Review of physical memory and memory management hardware
2. Overlays, swapping, and partitions
3. Paging and virtual memory
4. Working sets and thrashing
5. Caching

Competencies:

1. Explain memory hierarchy and cost-performance trade-offs.
2. Explain the concept of virtual memory and how it is realized in hardware and software.
3. Summarize the principles of virtual memory as applied to caching and paging.
4. Evaluate the trade-offs in terms of memory size (main memory, cache memory, auxiliary memory) and processor speed.
5. Defend the different ways of allocating memory to tasks, citing the relative merits of each.
6. Describe the reason for and use of cache memory
7. Compare and contrast paging and segmentation techniques..
8. Discuss the concept of thrashing, both in terms of the reasons it occurs and the techniques used to recognize and manage the problem.
9. Analyze the various memory portioning techniques including overlays, swapping, and placement and replacement policies.

OS / Device Management [elective]

Topics:

1. Characteristics of serial and parallel devices
2. Abstracting device differences
3. Buffering strategies
4. Direct memory access
5. Recovery from failures

Competencies:

1. Explain the key difference between serial and parallel devices and identify the conditions in which each is appropriate.
2. Identify the relationship between the physical hardware and the virtual devices maintained by the operating system.
3. Explain buffering and describe strategies for implementing it.
4. Differentiate the mechanisms used in interfacing a range of devices (including hand-held devices, networks, multimedia) to a computer and explain the implications of these for the design of an operating system.
5. Describe the advantages and disadvantages of direct memory access and discuss the circumstances in which its use is warranted.
6. Identify the requirements for failure recovery.
7. Implement a simple device driver for a range of possible devices.

OS / Security And Protection [core]

Topics:

1. Overview of system security
2. Policy/mechanism separation
3. Security methods and devices
4. Protection, access control, and authentication
5. Backups

Competencies:

1. Defend the need for protection and security, and the role of ethical considerations in computer use.
2. Summarize the features and limitations of an operating system used to provide protection and security.
3. Explain the mechanisms available in an OS to control access to resources.
4. Carry out simple sysadmin tasks according to a security policy, for example creating accounts, setting permissions, applying

OS / File Systems [elective]

Topics:

1. Files: data, metadata, operations, organization, buffering, sequential, nonsequential
2. Directories: contents and structure
3. File systems: partitioning, mount/unmount, virtual file systems
4. Standard implementation techniques
5. Memory-mapped files
6. Special-purpose file systems
7. Naming, searching, access, backups

Competencies:

1. Summarize the full range of considerations that support file systems.
2. Compare and contrast different approaches to file organization, recognizing the strengths and weaknesses of each.
3. Summarize how hardware developments have lead to changes in our priorities for the design and the management of file systems.

OS / Real Time and Embedded Systems [elective]

Topics:

1. Process and task scheduling
2. Memory/disk management requirements in a real-time environment
3. Failures, risks, and recovery
4. Special concerns in real-time systems

Competencies:

1. Describe what makes a system a real-time system.
2. Explain the presence of and describe the characteristics of latency in real-time systems.
3. Summarize special concerns that real-time systems present and how these concerns are addressed.

OS / Fault Tolerance [elective]

Topics:

1. Fundamental concepts: reliable and available systems
2. Spatial and temporal redundancy
3. Methods used to implement fault tolerance
4. Examples of reliable systems

Competencies:

1. Explain the relevance of the terms fault tolerance, reliability, and availability.
2. Outline the range of methods for implementing fault tolerance in an operating system.
3. Explain how an operating system can continue functioning after a fault occurs.

OS / System Performance Evaluation [elective]

Topics:

1. Why system performance needs to be evaluated
2. What is to be evaluated
3. Policies for caching, paging, scheduling, memory management, security, and so forth
4. Evaluation models: deterministic, analytic, simulation, or implementation-specific
5. How to collect evaluation data (profiling and tracing mechanisms)

Competencies:

1. Describe the performance measurements used to determine how a system performs.
2. Explain the main evaluation models used to evaluate a system.

OS / Scripting [elective]

Topics:

1. Scripting and the role of scripting languages
2. Basic system commands
3. Creating scripts, parameter passing
4. Executing a script
5. Influences of scripting on programming

Competencies:

1. Summarize a typical set of system commands provided by an operating system.
2. Demonstrate the typical functionality of a scripting language, and interpret the implications for programming.
3. Demonstrate the mechanisms for implementing scripts and the role of scripts on system implementation and integration.
4. Implement a simple script that exhibits parameter passing.

OS / Digital Forensics [elective]

Topics:

1. Digital forensics and its relationship to other forensic disciplines
2. Incident response responsibilities
3. Forensic procedures
4. Digital evidence and tracking
5. Rules/Standards of Evidence
6. Evidence gathering and analysis
7. Forensic mechanisms
8. Profiling
9. Tools to support investigative work

Competencies:

1. To explain the problems addressed by digital forensics and to outline the basic principles involved in its practice
2. To outline the basic processes of information gathering and analysis in line with best practices in digital forensics
3. To recognize the role that tools can play in digital forensics and to demonstrate their use in simple examples the use of tools and to demonstrate their use
4. To explain what questions must be asked and answered to determine if the interpretation of forensic evidence is valid or questionable.

OS / Security Models [elective]

Topics:

1. Models of protection
2. Memory protection
3. Encryption
4. Recovery management
5. Types of access control: mandatory, discretionary, originator-controlled, role-based
6. Access control matrix model
7. Harrison-Russo-Ullman model and undecidability of security
8. Confidentiality models such as Bell-LaPadula
9. Integrity models such as Biba and Clark-Wilson
10. Conflict of interest models such as the Chinese Wall

Competencies:

1. Compare and contrast current methods for implementing security.
2. Compare and contrast the strengths and weaknesses of two or more currently popular operating systems with respect to security.
1. Compare and contrast the security strengths and weaknesses of two or more currently popular operating systems with respect to recovery management.
2. Describe the access control matrix and how it relates to ACLs and C-Lists
3. Apply Biba's model to the checking of inputs in a program (tainted v. untainted, for example)
4. Describe how the Bell-LaPadula model combines mandatory and discretionary access control mechanisms, and explain the lattice formulation of Bell-LaPadula and Biba
5. Compare and contrast two security models
6. Relate particular security models to the models of the software life cycle
7. Apply particular models to different environments and select the model that best captures the environment

Appendix H: Competencies for Preparation in Databases

The Association for Information Systems (AIS) specifies coverage of a broad variety of topics in Data and Information Management in a bachelor degree program in Computer Science and Information Systems. Their suggested topics are show below.

AIS Curriculum for Data and Information Management

1. Database approach
2. Types of database management systems
3. Basic file processing concepts
4. Physical data storage concepts
5. File organizations techniques
6. Conceptual data model
 - a. Entity-relationship model
 - b. Object-oriented data model
 - c. Specific modeling grammars
7. Logical data model
 - a. Hierarchical data model
 - b. Network data model
 - c. Relational data model
 - i. Relations and relational structures
 - ii. Relational database design
8. Mapping conceptual schema to a relational schema
9. Normalization
10. Physical data model
 - a. Indexing
 - b. Data types
11. Database languages
 - a. SQL: DDL, DML, and DCL
12. Data and database administration
13. Transaction processing
14. Using a database management system from an application development environment
15. Use of database management systems in an enterprise system context
16. Data / information architecture
17. Data security management
 - a. Basic data security principles
 - b. Data security implementation
18. Data quality management
 - a. Data quality principles
 - b. Data quality audits
 - c. Data quality improvement
19. Business intelligence
 - a. On-line analytic processing
 - b. Data warehousing
 - c. Data mining
 - d. Enterprise search

The Association for Computing Machinery (ACM) has a similar set of topics for coverage in the core of a Computer Science bachelor's degree as shown in the following list:

IM/Information Models [core]

1. Information storage and retrieval (IS&R)
2. Information management applications
3. Information capture and representation
4. Metadata/schema association with data
5. Analysis and indexing
6. Search, retrieval, linking, navigation
7. Declarative and navigational queries
8. Information privacy, integrity, security, and preservation
9. Scalability, efficiency, and effectiveness
10. Concepts of Information Assurance (data persistence, integrity)

IM/Database Systems [core]

1. History and motivation for database systems
2. Components of database systems
3. DBMS functions
4. Database architecture and data independence
5. Use of a declarative query language

IM/Data Modeling [core]

1. Data modeling
2. Conceptual models (such as entity-relationship or UML)
3. Object-oriented model
4. Relational data model
5. Semistructured data model (expressed using DTD or XMLSchema, for example)

Coverage of the entire list for either organization typically requires more than one course in database technologies. Students coming into a bachelor's program with an associate degree are unlikely to have taken more than one database course. Our proposed competencies, therefore, summarize the most significant topics that would appear in introductory database design and implementation coursework.

Learning Objectives for Introductory Coursework in Database Design and Implementation

Upon completion of the course, students should be able to:

1. Articulate issues in Data and Information Management with respect to security, privacy, data integrity, performance, concurrency and data independence and discuss how database management systems address these issues.
2. Define the following logical database models: relational, object-oriented, hierarchical, file-based (from a historical perspective).
3. Demonstrate the ability to analyze a business problem and develop data management requirements.
4. Develop a conceptual model to meet the problem requirements using an Entity- Relationship diagram or another conceptual modeling tool (UML).
5. Apply normalization to reduce/eliminate redundancy in database design.
6. Implement the conceptual model in a commercial relational database such as MS Access, Oracle, MySQL, SQLServer, etc.
7. Demonstrate the ability to write Structured Query Language (SQL) statements for data definition, data manipulation and data control.
8. Design and implement forms, queries and reports in a commercial database.
9. Design and implement triggers and stored procedures in a commercial database.

Appendix I: Competencies for Preparation in Networks

Net Centric Computing [core]

Topics:

1. Background and history of networking and the Internet
2. Network architectures
3. The range of specializations within net-centric computing
4. Networks and protocols
5. Networked multimedia systems
6. Distributed computing
7. Client/server and Peer to Peer paradigms
8. Mobile and wireless computing

Competencies:

1. Discuss the evolution of early networks and the Internet.
2. Demonstrate the ability to use effectively a range of common networked applications including e-mail, telnet, FTP, wikis, and web browsers, online web courses, and instant messaging.
3. Explain the hierarchical, layered structure of a typical network architecture.
4. Describe emerging technologies in the net-centric computing area and assess their current capabilities, limitations, and near-term potential

NC/Network Communication [core]

Topics:

1. Network standards and standardization bodies
2. The ISO 7-layer reference model in general and its instantiation in TCP/IP
3. Overview of Physical and Data Link layer concepts (framing, error control, flow control, protocols)
4. Data Link layer access control concepts
5. Internetworking and routing (routing algorithms, internetworking, congestion control)
6. Transport layer services (connection establishment, performance issues, flow and error control)

Competencies:

1. Discuss important network standards in their historical context.
2. Describe the responsibilities of the first (lowest) four layers of the ISO reference model.
3. Explain how a network can detect and correct transmission errors.
4. Explain how a packet is routed over the Internet.
5. Install a simple network with two clients and a single server using standard host configuration software tools such as DHCP.

NC/Network Security [core]

Topics:

1. Fundamentals of cryptography
2. Secret-key algorithms
3. Public-key algorithms
4. Authentication protocols
5. Digital signatures
6. Examples
7. Network attack types: Denial of service, flooding, sniffing and traffic redirection, message integrity attacks, identity hijacking, exploit attacks (buffer overruns, Trojans, backdoors), inside attacks, infrastructure (DNS hijacking, route blackholing, misbehaving routers that drop traffic), etc.)
8. Use of passwords and access control mechanisms
9. Basic network defense tools and strategies
10. Intrusion Detection
11. Firewalls
12. Detection of malware
13. Kerberos
14. IPSec
15. Virtual Private Networks

16. Network Address Translation
17. Network Resource Management policies
18. Auditing and logging

Competencies:

1. Describe the enhancements made to IPv4 by IPSec
2. Identify protocols used to enhance Internet communication, and choose the appropriate protocol for a particular case.
3. Understand Intrusions and intrusion detection
4. Discuss the fundamental ideas of public-key cryptography.
5. Describe how public-key cryptography works.
6. Distinguish between the use of private- and public-key algorithms.
7. Summarize common authentication protocols.
8. Generate and distribute a PGP key pair and use the PGP package to send an encrypted e-mail message.
9. Summarize the capabilities and limitations of the means of cryptography that are conveniently available to the general public.
10. Describe and discuss recent successful security attacks.
11. Summarize the strengths and weaknesses associated with different approaches to security.

NC/Web Organization [Elective]

Topics:

1. Web technologies
 - a. Server-side programs
 - b. Client-side scripts
 - c. The applet concept
2. Characteristics of web servers
 - a. Handling permissions
 - b. File management
 - c. Capabilities of common server architectures
3. Role of client computers
4. Nature of the client-server relationship
5. Web protocols
6. Support tools for web site creation and web management
7. Developing Internet information servers
8. Publishing information and applications
9. Grid Computing, cluster, mesh
10. Web Services, Web 2.0, ajax

Competencies:

1. Explain the different roles and responsibilities of clients and servers for a range of possible applications.
2. Select a range of tools that will ensure an efficient approach to implementing various client-server possibilities.
3. Design and build a simple interactive web-based application (e.g., a simple web form that collects information from the client and stores it in a file on the server, and a server process to respond to the form data and produce a result)

NC/Networked Applications [elective]

Topics:

1. Protocols at the application layer
2. Web interfaces: Browsers and APIs
3. Web Search Technologies
4. Principles of web engineering
5. Database-driven web sites
6. Remote procedure calls (RPC)
7. Lightweight distributed objects
8. The role of middleware
9. Support tools
10. Security issues in distributed object systems
11. Enterprise-wide web-based applications
12. Service-oriented Architectures

Competencies:

1. Illustrate how interactive client-server web applications of medium size can be built using different types of Web technologies.
2. Demonstrate how to implement a database-driven web site, explaining the relevant technologies involved in each tier of the architecture and the accompanying performance tradeoffs.
3. Illustrate the current status of Web search effectiveness.
4. Implement an application that invokes the API of a web-based application.
5. Implement a distributed system using any two distributed object frameworks and compare them with regard to performance and security issues.
6. Discuss security issues and strategies in an enterprise-wide web-based application.

NC/Network Management [elective]

Topics:

1. Overview of the issues of network management
2. Use of passwords and access control mechanisms
3. Domain names and name services
4. Issues for Internet service providers (ISPs)
5. Security issues and firewalls
6. Quality of service issues: performance, failure recovery

Competencies:

1. Explain the issues for network management arising from a range of security threats, including viruses, worms, Trojan horses, and denial-of-service attacks
2. Summarize the strengths and weaknesses associated with different approaches to security.
3. Develop a strategy for ensuring appropriate levels of security in a system designed for a particular purpose.
4. Implement a network firewall.

NC/Compression [Elective]

Topics:

1. Analog and digital representations
2. Encoding and decoding algorithms
3. Lossless and lossy compression
4. Data compression: Huffman coding and the Ziv-Lempel algorithm
5. Audio compression and decompression
6. Image compression and decompression
7. Video compression and decompression
8. Performance issues: timing, compression factor, suitability for real-time use

Competencies:

1. Summarize the basic characteristics of sampling and quantization for digital representation.
2. Select, giving reasons that are sensitive to the specific application and particular circumstances, the most appropriate compression techniques for text, audio, image, and video information.
3. Explain the asymmetric property of compression and decompression algorithms.
4. Illustrate the concept of run-length encoding.
5. Illustrate how a program like the UNIX compress utility, which uses Huffman coding and the Ziv-Lempel algorithm, would compress a typical text file.

NC/Network Management [elective]

Topics:

1. Sound and audio, image and graphics, animation and video
2. Multimedia standards (audio, music, graphics, image, telephony, video, TV)
3. Capacity planning and performance issues
4. Input and output devices (scanners, digital camera, touch-screens, voice-activated)
5. MIDI keyboards, synthesizers
6. Storage standards (Magneto Optical disk, CD-ROM, DVD)
7. Multimedia servers and file systems
8. Tools to support multimedia development

Competencies:

1. For each of several media or multimedia standards, describe in non-technical language what the standard calls for, and explain how aspects of human perception might be sensitive to the limitations of that standard.
2. Evaluate the potential of a computer system to host one of a range of possible multimedia applications, including an assessment of the requirements of multimedia systems on the underlying networking technology.
3. Describe the characteristics of a computer system (including identification of support tools and appropriate standards) that has to host the implementation of one of a range of possible multimedia applications.
4. Implement a multimedia application of modest size.

NC/Mobile Computing [elective]

Topics:

1. Overview of the history, evolution, and compatibility of wireless standards
2. The special problems of wireless and mobile computing
3. Wireless local area networks and satellite-based networks
4. Wireless local loops
5. Mobile Internet protocol
6. Mobile aware adaption
7. Extending the client-server model to accommodate mobility
8. Mobile data access: server data dissemination and client cache management
9. Software package support for mobile and wireless computing
10. The role of middleware and support tools
11. Performance issues
12. Emerging technologies

Competencies:

1. Describe the main characteristics of mobile IP and explain how differs from IP with regard to mobility management and location management as well as performance.
2. Illustrate (with home agents and foreign agents) how e-mail and other traffic is routed using mobile IP.
3. Implement a simple application that relies on mobile and wireless data communications.
4. Describe areas of current and emerging interest in wireless and mobile computing, and assess the current capabilities, limitations, and near-term potential of each.

ADDENDUM

GENERAL STATEWIDE PROGRAM-TO-PROGRAM ARTICULATION in PENNSYLVANIA

WHEREAS, the General Assembly of the Commonwealth of Pennsylvania enacted Act 114 of 2006, which added to the Public School Code of 1949, Article XX-C entitled “Transfers of Credits Between Institutions of Higher Education” (referred to in this Agreement as the “Statewide Transfer System”);

WHEREAS, Act 114 of 2006 requires all community colleges in Pennsylvania and Pennsylvania State System of Higher Education (PASSHE) universities to participate in the Statewide Transfer System;

WHEREAS, Act 114 of 2006 permits independent and state-related institutions of higher education in Pennsylvania, as each is defined in Article XX-C, to elect to participate in the Statewide Transfer System;

WHEREAS, the General Assembly of the Commonwealth of Pennsylvania enacted Act 50 of 2009, which requires institutions participating in the Statewide Transfer System to accept the transfer of Associate of Arts and Associate Science degrees into parallel baccalaureate programs and recognize all competencies attained within the associate degree program;

WHEREAS, Act 50 of 2009 defines an Associate of Arts (AA) or Associate of Science (AS) degree containing a minimum of 60 college-level credits and designed primarily for transfer to a baccalaureate institution;

WHEREAS, Act 50 of 2009 requires the Transfer Articulation Oversight Committee (TAOC), as established in section 2004-C of the Public School Code of 1949, to identify Associate of Arts and Associate of Science degree programs for transfer with full junior standing into parallel baccalaureate degrees annually; and,

WHEREAS, Act 50 of 2009 requires members of the Transfer Articulation Oversight Committee established in section 2004-C of the Public School Code of 1949, to identify modifications that may be required in existing associate or baccalaureate degrees to satisfy external accreditation or licensure requirement;

All Institutions participating in the Statewide Transfer System enter into this Articulation Agreement and mutually agree as follows:

1. The statewide program-to-program articulation agreement ensures that students who complete an AA or AS degree from a participating institution will have their coursework and credits transfer into a parallel baccalaureate program with full junior standing and without the need for course-by-course equivalency.
2. Students are subject to the admissions and transfer credit policies of the participating institutions. The admissions and transfer credit policies for all of the institutions participating in Pennsylvania’s college credit transfer system can be found at www.PAcollegetransfer.com.
3. The AA or AS degree must include a minimum of 60 college-level credits designed and acceptable for transfer, not including developmental or remedial courses or career, technical or applied courses.
4. The transfer of coursework with a grade less than a C (2.0 on a 4.0 scale) in the AA or AS will be consistent with the policies of native students at the participating college or university.
5. Students and institutional personnel will be able to find out which institutions offer articulated programs by accessing a searchable database located at www.PAcollegetransfer.com. PDE will maintain this database through program information provided to TAOC by the individual participating institutions.
6. **Responsibilities of Associate Degree Institutions**
 - a. The AA or AS degree leading to a parallel bachelor degree will include the minimum number of credits and competencies of major-specific coursework as defined by the Agreement.
 - b. The AA or AS degree will meet the minimum requirements of the Commonwealth’s Transfer Credit Framework (“Framework”), as defined by the Statewide Transfer System.

- c. Any remaining AA or AS degree requirements will be accepted from arts and sciences electives designed and acceptable for transfer, not including developmental, remedial, career, technical or applied courses.
- d. By awarding the AA or AS, the Associate Degree Institution is validating that the student has met the competency requirements outlined in the Agreement.

7. Responsibilities of Bachelor Degree Institutions

- a. The Bachelor Degree Institution will recognize all competencies attained within the AA or AS degree and accept a transfer student who has earned the associate degree with full junior standing into a parallel baccalaureate degree program.
- b. All decisions made with respect to the transfer process shall be based on the principle of equivalence of expectations and requirements for native and transfer students.
- c. A transfer student's admission into the parallel baccalaureate degree will be subject to the Bachelor Degree Institution's specific requirements for admission to that major and be consistent with such requirements for native students.

8. Agreement Revision and Assessment

- a. Once a statewide program-to-program articulation agreement has been approved by TAOC, no amendments to the agreement can be offered by any party within the initial six (6) months of the agreement. After that time, a TAOC member with a proposed amendment to an approved agreement should submit the change to PDE.

Amendments that are offered as clarifying or technical but do not alter the substantive portions or intent of the agreement must be forwarded to TAOC. TAOC representatives will have at least thirty (30) days to review, comment and approve or deny the proposed amendments.

Amendments that seek to alter the substantive nature or intent of the agreement in any part must be forwarded to the appropriate PAC for review and consideration. The PAC will then make a recommendation to the TAOC, and TAOC shall approve or deny the proposed amendments.²

- b. PDE and TAOC will exercise responsibility for monitoring the effectiveness of the Agreement and its implementation.
- c. PDE shall collect data annually from the participating institutions that will enable the Department and TAOC to assess the effectiveness of the implementation of the Agreement in fostering a seamless transfer process and the academic success of transfer students at the senior institutions.

9. Transfer Appeal Process

- a. In accordance with Pennsylvania's Statewide Transfer System, each Bachelor Degree Institution shall have a procedure through which a transfer student can appeal a decision that he/she believes is not consistent with this Agreement.
- b. The Transfer Appeal Process shall be published, at minimum, in the institution's catalog and posted to the Commonwealth's official website of the Statewide Transfer System, www.PAcollegetransfer.com.

10. Institutional Resolution of Disputes

- a. In the event that an Associate Degree Institution considers the decision of a Bachelor Degree Institution to be inconsistent with this Agreement, the Associate Degree Institution shall consult directly with the Bachelor Degree Institution and attempt to resolve the matter.
- b. If the institutions are unable to resolve the issue, the Associate Degree Institution may submit their concern to PDE for consideration by the TAOC Dispute Resolution Committee. The Dispute Resolution Subcommittee will act according to the policies and procedures developed by TAOC as part of the Statewide Transfer System. The determination made by the Dispute Resolution Subcommittee will be binding upon the parties.

² Approved by TAOC and added to agreement on August 18, 2011.

11. Implementation Date and Applicability

Having fulfilled the requirements outlined in the Program-to-Program Articulation Agreement, students transferring with an AA or AS degree from a participating institution will be considered by the receiving baccalaureate degree institution to have received adequate preparation in the field of study at the foundation level and therefore eligible to transfer as a junior into advanced major coursework.

Participating institutions will enact the Agreement in accordance to the timeline outlined by the TAOC, but no later Fall 2013.³

Continuation of the agreement remains in effect until such time as all cooperating institutions of the Statewide Transfer System formally approve any revisions.

GLOSSARY OF TERMS

Articulation: The aligning of curriculum between institutions of higher education to ensure the efficient and effective movement of students among those institutions.

Associate of Arts (AA) and Associate of Science (AS) Degree: A degree consisting of at least 60 college-level credits and designed for transfer into a baccalaureate degree program.

Foundation Coursework: Courses at a level of comprehension usually associated with freshman and sophomore students and typically offered during the first half of a baccalaureate degree program. Such coursework typically does not have course prerequisites.

Native Student: A student who entered a given college or university without first matriculating at another college.

Parallel Baccalaureate Degree: A bachelor degree program in a comparable field of study and with similar foundation-level major-specific competencies as an associate degree program.

Receiving Institution: The college or university where a transfer student plans to enroll and to apply previously earned credit toward a degree program.

Transfer Credit: The credit granted by a college or university for college-level courses or other academic work completed at another institution.

Transfer Student: A student who enters a participating college or university after earning college-level credit at another college or university.

Transfer: The process by which a student moves from one postsecondary institution to another. Also refers to the mechanics of credit, course and curriculum exchange between institutions.

Advanced Coursework: Courses with advanced depth of content knowledge in the field of study and carry the expectation of more complex competencies identified in the expected student learning outcomes is referred to as advanced coursework. These courses often have prerequisites and are usually beyond the "Introduction to..." or "Foundation of..." level.

³ Agreements approved by TAOC prior to August 31, 2011 must be implemented by the institutions by Fall 2012. Agreements approved by TAOC after August 31, 2011 but before May 1, 2012 must be implemented by the institutions by Fall 2013.