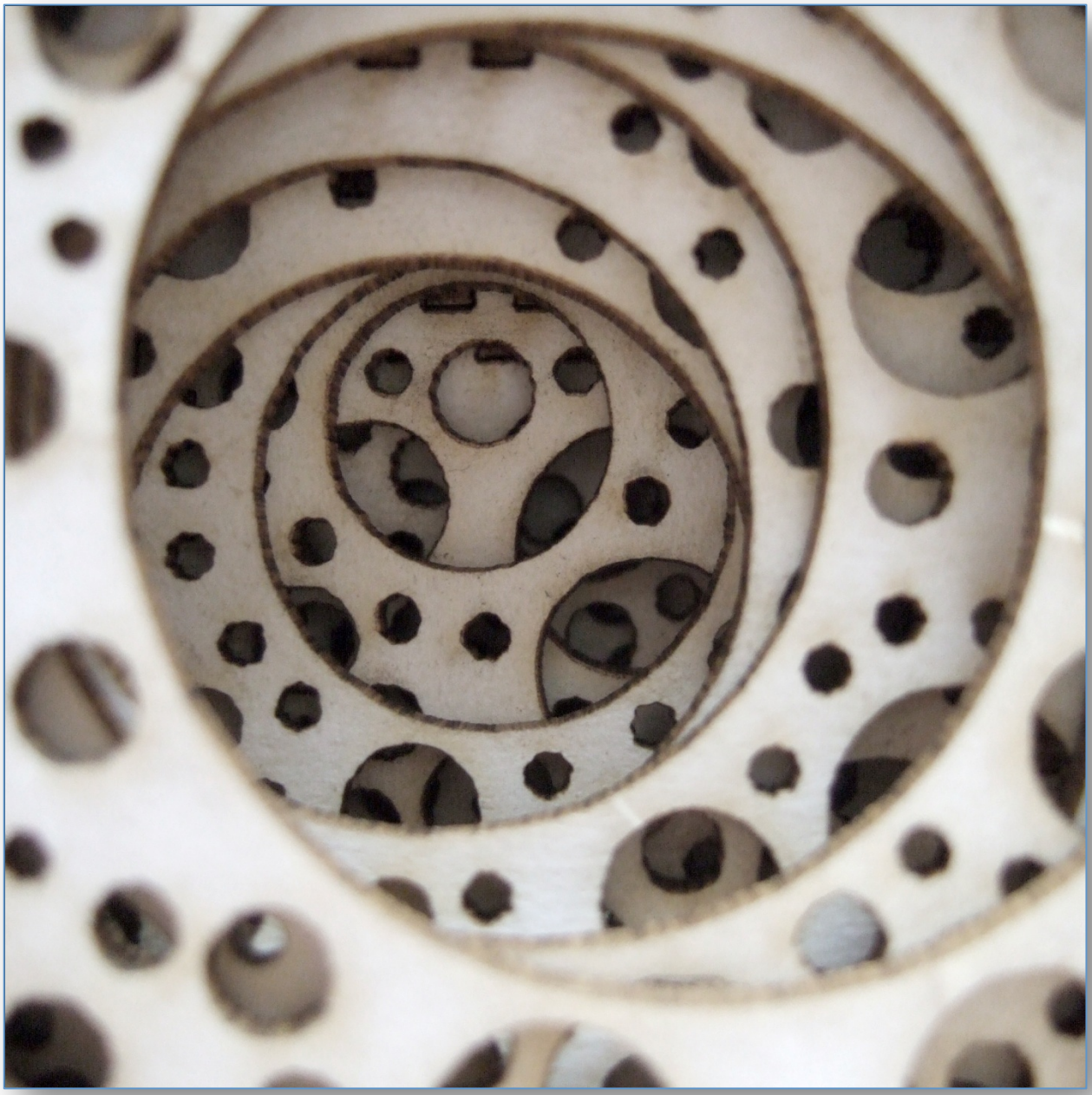

Computer Science Course Learning Outcomes

This document contains learning outcomes for courses that are offered by the Department of Computer Science at The University of Wisconsin La Crosse

Computer Science Department
Spring, 2011

Table of Contents

CT 100 - Introduction to Computational Thinking

Catalog Description

Computational thinking represents a universally applicable collection of concepts and techniques borrowed from computer scientists. This course is designed to teach how to think algorithmically; to examine the ways that the world's information is encoded and how this impacts our lives; to explore the capabilities and limitations of computers from the past, the present and the future; to apply software design diagramming techniques to model real-world systems; to learn how the rules of logic apply to computation, reasoning and discourse; to examine how computers both enhance and constrain our lives; to explore many of the problem solving strategies used by software developers and how they are useful to you.

Prerequisites:

None.

Learning Objectives

Students shall be able to:

- General Computation
    - be familiar with the history of the modern digital computer from the 1950's
    - be able to apply Moore's law as a measure of computing advances
    - understand the limitations and advantages of digital devices by comparison to analog
    - appreciate the computer as a repository of data
    - understand that data is represented as sequences of bits
    - know the relative sizes of byte, kilobyte, megabyte, terabyte and petabyte
    - be able to translate decimal numbers to binary and binary to decimal
    - understand that data has alternative forms (numeric, textual, code, image, sound)
    - understand that different data forms use different, largely incompatible encodings
    - be familiar with the concept of data compression and the difference between lossy and nonlossy compression
    - understand different forms of computer storage and the relative size and performance differences between memory and secondary storage
    - understand the process of program compilation and execution (source code and executables)
- Logic
    - understand that one bit of data may also store a Boolean value (true or false)
    - be able to analyze a Boolean expression involving the operations of NOT, AND, OR, IMPLIES and EQUIVALENCE
    - be able to express factual English sentence in the form of propositional logic
    - be familiar with logical expression applications in expressing software requirements
    - be able to express database queries at the level of propositional logic
- Problem Solving
    - understand the nature of software development as a problem solving activity
    - be able to apply divide and conquer strategies to basic problems

- c be able to develop story boarding for solving real world scenarios
    - understand the problem solving approach of prototyping and be able to apply prototyping to implementing simple sequential algorithms
    - be able to translate simple story boards into algorithms involving objects and a solution as a sequence of method calls
- Control
    - understand the concept of software and program execution
    - be able to write simple programs involving sequences of methods applied to different objects
    - understand that algorithms involve choices and choices take the form of selections involving logical conditions
    - understand that algorithms often involve repetition
    - understand how algorithms are modularized
- Models of Computation
    - know the definitions of basic digraph terminology, including nodes, vertices, arcs, cycles, walks, and paths.
    - understand the basics of using graphical abstractions as models of computation
    - know the form and function of flow chart elements for imperative statements, selection and repetition
    - be able to express simple algorithms using flow charts
    - understand the concepts of computational state, events and actions
    - be able to model simple, sequence algorithms of ten or less states
- Organizing Data
    - be able to select meaningful names and understand why this is importance for such things as URLs, files, variables, and so forth.
    - understand the role of unique naming and the concept of case sensitivity
    - be able to craft path names for files organized into file trees
    - understand the concept of direct access and indexed data retrieval
    - understand the concept of sequential, linked, data retrieval
    - know the difference between linear and non-linear data organizations
    - understand the concept of data organized in tree structures and applications in classification and analysis.
    - understand the concept of 2-dimensional data layout and tabular information retrieval
- Code is Data / Data is code
    - understand the von Neumann principle that computers can be used to store both data and code
    - be able to craft simple spreadsheet
    - understand the concept of 2-dimensional data layout and tabular information retrieval
    - understand the relationship of discrete functions and tables
    - appreciate the role of patterns in software development and the use of software for pattern matching of strings

- Performance and Limits of Computation
    - recognize that performance can be characterized by a count of operations that are performed
    - understand the distinct between best case and worst case analysis and the importance of the latter
    - understand the limitations of benchmarking vs. analytic analysis
    - to be able recognize the difference between tractable (polynomial) and brute force (exponential)
    - understand that many security mechanisms rely upon algorithmic performance
    - recognize that not all things are actually computable (Halting problem)
- Verification and Validation
    - to understand that software correctness only makes sense with respect to specifications
    - to be able to analyze a small programming problem, the outcome of which is a set of software requirements
    - to understand that proving program correctness is sometimes a possibility, but not often a reality.
    - to understand that software correctness is usually accomplished by way of software tests
    - to realize the limitations of software testing
    - to be able to craft test cases using simple boundary condition analysis
- Concurrency
    - to understand that software correctness only makes sense with respect to specifications
    - to understand the distinction between sequential and concurrent execution.
    - to learn how sorting can be accomplished concurrently with a sorting network.
    - to understand the problem of data integrity in a concurrent environment and some common solutions to data corruption avoidance.
    - to understand the potential for deadlock and strategies for avoiding deadlock.
- Security
    - to understand the basic tenants of security - integrity, confidentiality, and availability.
    - to realize that complete mitigation is rarely possible, but that a certain measure of security is possible using mechanisms for deterrence, deflection and detection/recovery.
    - to know basic security terminology, including vulnerability, asset, security system, mitigation, virus, spoofing, spam, sniffing, DOS and CERT.
    - to understand the proper use of security mechanisms including anti-virus software, passwords, encryption, file backup and firewalls.
    - to understand the potential for deadlock and strategies for avoiding deadlock.

CS 101 - Introduction to Computing

Catalog Description

Computers and computer software are an integral part of modern society. This course explores this relationship. Students will examine the computer as a problem-solving tool through the use of database, spreadsheets and small scale programming. Students will examine the computer as a communication tool through the use of word processing and the Internet. Other topics include the history and future of computer technology, computer hardware basics, man/machine relationships, applications of computers in various disciplines, and social/ethical issues. Credits earned in CS 101 cannot be applied to the CS major or minor.

Prerequisites:

None.

Learning Objectives

Students shall be able to:

- Describe the function of a CPU
- Describe and characterize various types of memory including: cache, RAM, and magnetic disk.
- Read and create imperative programs that include the following constructs: assignment, looping, conditionals and procedure calls
- Read and create spreadsheets that include: function calls, expressions and charts
- Describe how computers are used across a wide variety of academic and professional disciplines.
- Describe issues of computer security including: phishing, sniffing, encryption, biometrics, and authentication schemes.

CS 102 - Web Development and the Internet

Catalog Description

An introduction to the Internet and the fundamentals of Web page design including history and technical structure of the Internet, markup languages and Web page programming. Technical issues include client/server interfaces and network protocols. Other topics include societal issues such as privacy, security, accessibility, intellectual property, and censorship.

Prerequisites:

CS 101 or CS 120.

Learning Objectives

Students shall be able to:

- Read and construct HTML documents.
- Read and construct CSS documents
- Read and construct XML documents
- Describe the client/server architecture
- Describe the HTTP protocol
- Describe security issues as they relate to web-based communication
- Describe basic issues concerning intellectual property including: copyright, fair-use, DMCA, and case studies involving RIAA.

CS 202 – Introduction to Web Design and Development

Catalog Description

This course is an introduction to Web page design and website management. Students will be introduced to browser/server interaction, Web page design, techniques for creating media rich graphical interfaces, and ethical considerations regarding intellectual property rights and security. Students will receive hands-on experience in a variety of standardized Web technologies and coding languages to develop dynamic, functional, and appealing Web pages.

Prerequisites:

CT 100 or CS 120 or familiarity with some programming language

Learning Objectives

Students shall be able to:

- Understand, create and validate HTML/XHTML documents. Including the following constructs:
  - a, abbr, acronym, address, area, b, base, blockquote, body, br, button, caption, cite, col, group, dd, del, fn, div, dl, DOCTYPE, dt, em, fieldset, form, h1, h2, h3, h4, h5, h6, head, html, hr, i, img, input, ins, label, legend, li, link, meta, noscript, object, ol, optgroup, option, p, param, pre, script, select, small, span, strong, style, sub, sup, table, tbody, td, textarea, tfoot, th, thead, title, tr, tt, ul, var
- Understand and create XML documents.
- Understand and create CSS specifications.
- Understand and create JavaScript programs capable of client-side validation.
- Understand and apply basic elements of visual design to construct Web pages. Topics include color, typography, and layout.
- Understand the client/server architecture.
- Describe the basic elements and usage of the HTTP protocol.
- Describe security issues as they relate to web-based communication.
- Describe basic ethical and legal implications related to web development including copyright, security and accessibility.

CS 120 - Software Design I

Catalog Description

An introduction to the fundamentals of software development; including software classes, objects, inheritance, polymorphism, logic, selection control, repetition control, subprograms, parameter passage, and rudimentary software engineering techniques. Students complete numerous programming projects using a modern programming language.

Prerequisites:

MTH 151 or 175, or math placement test scores at, or above, MTH 151.

Learning Objectives

Students shall be able to:

- Write Java programs using non-parallel control instructions, including assignment, method call (void and non-void), if, while, for (iterative).
- compose basic expressions using literals, variables, and the following operations:
    - integer & real: - (negation), +, -, /, *, %, ++(prefix & postfix), --(prefix & postfix), =, !=, <, <=, >, >=
    - boolean: !, &&, ||
    - String: +
- Understand the precedence of such expressions with and without parentheses.
- Compose primitive expressions involving mixed types, widening, and casts.
- Write variable declarations, including final, and understand local, private, public and protected scope; and the use of the this notation.
- Understand the principle of information hiding and use it to select between scope alternatives.
- Compose instructions to assign objects, understanding the concept of object binding, including the null notation and orphan objects.
- Read and interpret syntax diagrams.
- Draw, interpret, and trace code using object diagrams.
- Draw and interpret class diagrams, including scope annotations and aggregation and inheritance relations.
- Develop programs involving all of the following algorithm patterns: variable content swap, cascading if instructions, counting loops, linear searches (lists and arrays), selection sort, insertion sort, object access shared by multiple classes, method callback.
- Interpret and utilize inheritance, overloading, overriding, including the super notation.
- Read method preconditions and postconditions using propositional logic.
- understand and correct code exhibiting infinite loops, NullPointerExceptions and ArrayIndexOutOfBoundExceptions
- Debug by inserting println instructions.
- Adhere to fundamental programming style conventions, including using meaningful identifiers, intelligent inclusion of comments and proper indentation patterns.
- Write code as a client of a singly-linked list class.

- Write both client using simple parameterized (generic) classes.
- Write code involving one-dimensional arrays.
- Write and interpret code with import declarations.
- use and interpret code involving the following standard Java classes: & methods: Object (equals, toString), String (+, length, charAt), Math (random, abs, sqrt, trigonmetry functions), JFrame, Container (add, remove, repaint), JFrame, JComponent (paint)
- Understand the purpose of wrapper classes, autoboxing and autounboxing.
- Write and read code involving event handling via inheritance and method overriding.

CS 220 - Software Design II

Catalog Description

This is a second course in the design of programs. Emphasis is placed on data abstraction and its application in design. Definitions of abstract data types are examined. The following structures are examined as methods for implementing data abstractions: recursion, sets, stacks, queues, strings, and various linked lists. Students will be expected to write several programs using these techniques in a modern programming language

Prerequisites:

CS 120.

Learning Objectives

Students shall be able to:

- Use a production quality IDE.
- Write code involving multi-dimensional arrays.
- Understand the separation of abstraction and implementation and choose between alternative linear container representations.
- Write and interpret preconditions, postconditions and class invariants.
- Write and read code involving exceptions, including try blocks, throw instructions, and throws qualifiers.
- Understand the relationship between files and directories in a hierarchical file directory system and name files with both relative and absolute names.
- Understand the distinction between binary and text files, select between them, and translate data of each type to the equivalent other type (including use of the Scanner class).
- Read and write code using the following classes (and associated methods): DataInputStream (close, read & write methods), DataOutputStream (flush, close, all read & write methods), BufferedReader (close, read, readLine), PrintWriter (close, print, println), and Scanner (next and hasNext methods).
- Read and write recursive definitions.
- Read and write Backus Naur Form definitions of syntax.
- Read and write recursive methods, both void and non-void.
- Design programs involving the template design pattern and use Java interfaces and abstract classes to implement this pattern.
- Trace the behavior of code using static, dynamic and automatic memory, understanding the usage of each.
- Read and write code implementing linked lists (both singly-linked and doubly-linked), including traversal, insertion and item removal algorithms, sentinel cells, and prepointers.
- Read and write code involving inner private classes and understand their utility for implementing linked lists in the style of java.util.LinkedList.
- Read and write code involving stacks and queues.
- Perform simple counting analyses on linear, polynomial and logarithmic algorithms.

- Develop programs involving all of the following algorithms: binary search, linear search (for both arrays and lists), merge sort, and quicksort.
- Interpret and utilize object type conformance and subtype polymorphism.

CS 224 - Introduction to the ? Programming Language

Catalog Description

This course presents the syntax and semantics of a particular programming language. Different offerings of the course will present different languages. Students are expected to be fluent in another programming language prior to enrollment.

Prerequisites:

A prior computer science course appropriate to the language being offered.

Learning Objectives:

C[1]

Students shall be able to:

- Explain the origins of the C language as a substitute for assembly language in developing the UNIX operating system.
- Understand the organization of a C program as a collection of subprograms (functions) that can be stored into separate files and compiled independently of each other.
- Describe the role of the C preprocessor.
- understand the syntax and semantics of the following elements of the C language:
  - data types and data conversions
  - operators and expressions
  - input/output operations through library functions
  - looping statements
  - conditional statements
  - other statements/operators peculiar to the language
  - pointers
  - records (*struct*s in C)
- Explain the concept of dynamically allocated data and its implementation in C: the library functions *malloc()* and *free()*.
- Understand the concepts of lists and linked lists and different ways of implementing them in C.
- Write and run numerous C language programs to demonstrate understanding of the topics listed above.

Python

Students shall be able to:

- Understand the syntax and semantics of the following elements of Python:
  - Simple data types including bool, int, float, complex
  - Collection data types including string, tuple, list, set, dictionary
  - Other built-in types including File, Function, Class, Method
  - Operators and expressions involving:
    - relational operators: <, >, <=, >=, ==, !=, <>, is, is not
    - arithmetic operators: +, -, /, *, %, **

[1] The specific learning objectives for this course will depend on the language. This document lists the SLO's for recent offerings.

        - logical operators: not, and, or
        - list comprehension
    - Control flow including loops and conditionals
- Understand how to express and manipulate regular expressions.
- Understand the difference between classes and objects.
- Write classes and create and use objects.
- Understand file I/O
- Understand and implement programs that involve the above topics

CS 270 - Assembler Programming and Introduction to Computer Organization

Catalog Description

Programming in machine and assembler language is integrated with an introduction to the organization of computer hardware. An examination of the instruction set merges with descriptions of the related hardware devices. Laboratory assignments include the construction of software, as well as hardware, units. Topics include basic instruction types, data representation, addressing modes, combinational circuit design, flip-flops, registers, the ALU, computer memory, and interrupt handling.

Prerequisites:

CS 120 and MTH 225 or concurrent enrollment.

Learning Objectives

Students shall be able to:

- Describe the basic organization of a von Neumann computer's hardware components.
- Understand the programming language hierarchy from high-level language to assembly language to machine language.
- Explain how data is encoded in a computer, including unsigned integers, signed integers, two's complement representation, and character codes.
- Demonstrate proficiency in conversions between different number bases, with an emphasis on binary.
- Describe different ways of defining machine language formats and number of operands.
- Understand the instruction set of a sample computer, including instructions for data movement, arithmetic and logical operations, and transfers of control.
- Describe a variety of addressing modes in machine-level instructions.
- Translate high-level language constructs into assembly language as a compiler would do.
- Understand the interplay of subroutines, the run-time stack, and parameters.
- Write and run numerous assembly language programs to demonstrate knowledge of the topics listed above.
- Describe how input/output operations are performed at the machine level.
- Explain the concept of interrupts, and write assembly language programs that perform both busy-wait I/O and interrupt-driven I/O.
- Describe the operation of logic gates and their composition into combinational circuits.
- Minimize digital circuits using Karnaugh maps.
- Understand sequential circuits (state elements), including latches and flip-flops, and the role of the system clock.
- Explain how main memory can be organized as a two-dimensional array of flip-flops or capacitors.
- Construct digital circuits in a laboratory setting with chips and breadboards.

CS 340 - Software Design III: Abstract Data Types

Catalog Description

An extensive survey of data structures and associated algorithms. An introduction to algorithm efficiency measures is included as a tool for deciding among alternate algorithms. Topics include: searching and sorting in arrays, hash tables, tree traversal and search algorithms, expression evaluation and graphs. Usually Offered: Fall and Spring Semester

Prerequisites:

CS 220 and MTH 225.

Learning Objectives

Students shall be able to:

- Give abstract descriptions (a listing of method signatures and associated data) of the following data types:
    - list
    - stacks
    - queues (including priority queues)
    - trees (binary search trees, and self-balancing binary search trees)
    - graphs
    - maps
- Implement each of the data types listed above using a variety of commonly utilized representations and algorithms. Specifically,
    - lists: singly and doubly linked as well as sequential implementation
    - stacks: linked and sequential
    - queues: linked, sequential including heaps
    - binary trees: linked and sequential
    - graphs: adjacency list and adjacency matrices
    - maps: hashtables
- Characterize the time and space complexity of algorithms and data structures using asymptotic notation.
- Characterize the time complexity of the following sorting routines:
    - heapsort
    - mergesort
    - quicksort
    - insertionsort
- Choose an efficient implementation of a data type for a particular application.

CS 341 - Software Design IV: Software Engineering

Catalog Description

A study of methodologies for the development of reliable software systems. Several specification, design, and testing techniques are surveyed with an emphasis on one particular formal specification and formal design technique. Students work in teams, applying these techniques to the development of a medium scale (2000-5000 lines) software product.

Prerequisites:

CS 340.

Learning Objectives

Students shall be able to:

- Specify a software system using a formal specification language such as Z or Object Z.
- Read and write UML class diagrams, UML sequence diagrams, UML use case diagrams.
- Read and write a requirements document, design document, and test document.
- Define the following lifecycle models: waterfall, spiral, rapid prototyping, incremental, agile
- Choose the appropriate lifecycle model for a particular software development effort.
- Learn various software life cycle models that include water fall model, prototyping models (rapid, incremental and evolutionary), spiral model, and agile model.
- Choose appropriate life cycle model for a given software development project.
- Develop requirements document, design document, code and test cases for a software project. Implement and demonstrate the project.
- Work in team setting.

CS 342 - Software Testing Techniques

Catalog Description

As the size and complexity of software projects have grown, so has the importance of ensuring program correctness. This course examines the issues of program testing, validation, and verification. Course projects require students to construct test data and to analyze the correctness of several software systems.

Prerequisites:

CS 340.

Learning Objectives

Students shall be able to:

- Understand different activities in the testing process and level of testing
- Learn various testing techniques and limitations
- Generate test data for several simple software applications
- Explore advanced testing techniques widely used in the industry including JUnit, Selenium testing, and Mutation Testing.
- Learn some basic testing metrics and available tools

CS 351 - Simulation

Catalog Description

Programming computers to play games and imitate activities of systems such as drive-in facilities, checkout lanes, and computer operations. Topics include tests of goodness of fit, random number generators, simulated sampling, queuing theory, analysis of systems to be simulated, construction and validation of simulation programs, and interpretation of results.

Prerequisites:

CS 220 and MTH 207.

Learning Objectives

Students shall be able to:

CS 352 - Computer Graphics and Scientific Visualization

Catalog Description

An introduction to the fundamentals of computer graphics and its application to Scientific Visualization. Topics include basic imaging algorithms, geometric transformations, hidden surface algorithms, polygonal rendering, solid modeling, lighting models, and specialized graphics hardware.

Prerequisites:

CS 340 and MTH 207.

Learning Objectives

Students shall be able to:

- Explain and solve problems related to the fundamental drawing algorithms (e.g. line drawing, triangle fill, depth buffer).
- Explain and solve problems related to the mathematics used in representing 2D and 3D space transformations.
- Describe the common lighting models used in computer graphics (e.g. ambient, diffuse, specular, Phong, Gourard).
- Demonstrate a working knowledge of OpenGL geometry based drawing.
- Demonstrate a working knowledge of OpenGL shader based drawing.
- Demonstrate a working knowledge of scene graph based systems.
- Demonstrate a working knowledge of web based 3D drawing standards (e.g. VRML,
- X3D).
- Demonstrate a working knowledge of ray tracing methods and volume visualization.
- Demonstrate a working knowledge of procedural shading.
- Demonstrate a working knowledge of data structures used to represent surface models.
- Describe the common applications of computer graphics to scientific visualization, augmented reality and virtual training environments.

CS 353 - Analysis of Algorithm Complexity

Catalog Description

An in-depth analysis of the computational complexity of a wide range of algorithms for problems of fundamental importance to computer science. Algorithms to be examined include: sorting, pattern matching and various graph algorithms.

Prerequisites:

CS 340 and MTH 207

Learning Objectives

Students shall be able to:

- Give the computational complexity of an algorithm using asymptotic notation.
- Design efficient algorithms for various types of problems.
- Analyze the resource complexity of an algorithm.
- Identify and distinguish between the following techniques: greedy, linear programming, dynamic programming, divide and conquer.
- Understand the difference between the computational complexity of a particular algorithm for solving a given problem, and the complexity of the problem itself.
- Explain NP-Completeness and deal with NP-complete problems.
- Apply classical sorting, searching, optimization and graph algorithms.

CS 364 - Introduction to Database Management Systems

Catalog Description

Introduction to the design and organization of database management systems. Topics include the relational data model, relational algebra, SQL query language, database software development, data security, normalization, client/server environments.

Prerequisites:

CS 220

Learning Objectives

Students shall be able to:

- Create an Entity Relationship (ER) Model and ER Diagram from a database problem statement
- Implement an ER model in a relational database using SQL
- Query a relational database using SQL
- Modify a relational database using SQL
- Write stored procedures for a relational database
- Write application programs that use embedded SQL to communicate with a relational database
- Write queries using relational algebra
- Understand the basics of query processing
- Identify functional dependences from a database problem statement
- Design a relational database that satisfies 3NF or BCNF
- Identify where database indexes can be used to improve query performance and implement appropriate indexes using SQL

CS 370 - Computer Architecture

Catalog Description

A presentation of the logical organization of modern digital computers. Topics include performance evaluation, instruction set design, computer arithmetic, processor control, pipe-lining, cache memory, memory hierarchy, memory and system buses, and I/O organization.

Prerequisites:

CS 270 and MTH 225.

Learning Objectives

Students shall be able to:

- Describe computer architecture and organization, computer arithmetic, and CPU design
- Identify high performance architecture design
- Use assembly language to program a microprocessor system
- Explain the organization of the von Neumann machine and its major functional units.
- Explain how an instruction is executed in a von Neumann machine.
- Understand instruction execution through instruction cycles
- Explain the basic concepts of interrupts and how interrupts are used to implement I/O control and data transfers.
- Explain the memory hierarchy
- Understand PCI bus traffic and how data transfers are performed
- Explain the reasons for using different formats to represent numerical data

CS 402 - Web Application Development

Catalog Description

This course will give a detailed description of the core concepts and general principles of Web application development. The course will cover various protocols, programming languages, scripting languages, data storage and security, layered software architectures, and graphical interface design as they relate to Web development.

Prerequisites:

CS 340 or consent of instructor.

Learning Objectives

Students shall be able to:

- Describe the HTTP protocol in detail.
- Understand and construct CSS specifications.
- Read and construct XML documents.
- Define how XML documents are validated and how they are commonly used.
- Understand and write JavaScript programs that exhibit an understanding of: the DOM; regular expressions; client-side validation; functions; loops; conditionals; dynamic web pages.
- Understand and write HTML pages including: forms; images; divs; spans; buttons; fieldsets and tables.
- Define cookies and describe common uses.
- Describe the basic elements of a Java based server architecture including: tomcat, JSP, JSF, JPA, and servlets.
- Implement a web application that exhibits: form entry, client-side validation of user input, themes, dynamic web pages, proper use of CSS and HTML, state management, secured data, and server-side persistence.

CS 410 - Free and Open Source Software Development

Catalog Description

This course will examine the history and scope of the Free and Open Source Software movement. The course will survey the various definitions of open source licenses and examples of major free and open source development projects (e.g. the GNU Project, Apache Software Foundation, Linux). The course will also examine the development tools that support developer communities as well as how the World Wide Web has created the possibility of international development teams. Students will select and contribute to the development of an existing open source project.

Prerequisites:

CS 340.

Learning Objectives

Students shall be able to:

- Discuss the historical precedents for "open" development engineering projects (e.g. BSD Unix, Internet Engineering Task Force).
- Discuss the legal concepts of copyright and patent.
- Discuss the key characteristics of free and open software licenses.
- Discuss the history and practice of software patents and their impact on free and open development.
- Discuss the common organizational forms of free and open development projects.
- Demonstrate a working knowledge of project hosting sites.
- Demonstrate a working knowledge of common source code management tools used in free and open development projects.
- Demonstrate a working knowledge of common development tools used in free and open development projects.

CS 421 - Programming Language Concepts

Catalog Description

A comparative study of the concepts underlying the design of contemporary high-level programming languages, including imperative, functional, logic and object-oriented paradigms; formal representation of syntax and semantics; control structures; data and procedural abstraction; scope and extent; parallelism and exception handling.

Prerequisites:

CS 340

Learning Objectives

Students shall be able to:

- Read and write a syntactic specification using BNF.
- Describe how the semantics of a language are specified, specifically addressing axiomatic and denotational semantics.
- Describe the core features of the functional, logical, object-oriented and imperative programming paradigms.
- Write and interpret programs using each of the paradigms listed above.
- Describe and characterize data types such as: primitives, compound types, arrays, enumerations, sets, closures, and pointers
- Describe language level constructs that support concurrency and contrast with mechanisms providing library support for concurrency
- Define and give examples in some implemented language of the following concepts:
    - strong and weak typing
    - type inference
    - static and dynamic typing
    - generics
    - dynamic dispatching
    - parameter passing (by reference, by value)
    - currying
    - macro
    - eager and lazy evaluation
    - subtyping and subclassing
    - algebraic type specification

CS 431 - Introduction to Robotics

Catalog Description

This course is a hands-on introduction to the algorithms and techniques required to write robot control software. Topics include the components of mobile robots and robot manipulators, manipulator kinematics, robot task planning, sensing, sensor fusion, visual servoing and robot control concepts.

Prerequisites:

CS 340.

Learning Objectives

Students shall be able to:

- Compare and contrast the hierarchical and reactive paradigms.
- Describe the biological foundations of robotic control including: reflexive, reactive and conscious animal behaviors; innate, innate with memory and learned; behavioral schema; behavioral conflict resolution including equilibrium, dominance and cancellation.
- Describe the sense-plan-act cycle of control
- Describe robotic configuration using transformation matrices.
- Describe robotic motion in configuration space and determine paths in configuration space.
- Perform path planning using the A* algorithm
- Give the major characteristics of sensors including: rotary encoders, sonar, radar, laser range finders, cameras and GPS.
- Represent maps using a topological or metric approach
- Describe the inverse and forward kinematic problems.
- Construct autonomous robots using the reactive paradigm to perform path following and obstacle avoidance
- Define the terms:
    - frame problem
    - closed world assumption
    - symbol grounding problem
    - manipulator
    - path planning
    - degrees of freedom
    - configuration space
    - actuator
    - sensor fusion
    - emergent behavior

CS 441 - Operating System Concepts

Catalog Description

The study of the structures and algorithms of operating systems. Operating systems are viewed as managers and controllers of resources such as processors, memory, input and output devices and data. Topics include multiprogramming systems, CPU scheduling, memory management and device management.

Prerequisites:

CS 340

Learning Objectives

Students shall be able to:

- Explain what constitutes an operating system and why it is needed.
- Describe the services an operating system provides to users, processes, and other systems.
- Identify the major components of an operating system.
- Understand the notion of a process (a program in execution) and how it forms the basis of all computation.
- Describe the various aspects of processes, including scheduling, creation and termination, and communication.
- Explain different approaches to communication in client-server systems.
- Know the difference between a thread and a process and to explain the benefits of multithreading.
- Explain various CPU-scheduling algorithms and compare them with respect to turnaround time, throughput, and risk of starvation.
- Explain process synchronization and the critical-section problem.
- Identify both software and hardware solutions to the critical-section problem.
- Understand the nature of deadlocks in a computer system.
- Identify and understand different methods for preventing, avoiding, and detecting deadlocks.
- Understand the major memory-management techniques: contiguous allocation, paging, and segmentation.
- Describe the benefits of a virtual memory system.
- Explain the concepts of demand paging, page-replacement algorithms, and allocation of page frames.
- Understand the principles of the working-set model.
- Explain the properties of file systems and their interfaces.
- Understand file-system design tradeoffs, including access methods, file sharing, protection, and directory structures.
- Discuss file block allocation and free-block algorithms.

CS 442 - Structure of Compilers

Catalog Description

A comparative study of the concepts underlying the design of contemporary high-level programming languages, including imperative, functional, logic and object-oriented paradigms; formal representation of syntax and semantics; control structures; data and procedural abstraction; scope and extent; parallelism and exception handling.

Prerequisites:

CS 340

Learning Objectives

Students shall be able to:

- Define deterministic and non-deterministic finite automata (FA)
- Covert non-deterministic FA to equivalent deterministic FA
- Define regular expressions (RE)
- Create RE to specify the tokens of a programming language
- Create finite automata to recognize a language specified by a RE
- Understand the characteristics of languages that cannot be specified by RE or recognized by FA
- Use a scanner generator to build a scanner for a programming language
- Define deterministic and non-deterministic push down automata (PDA)
- Define context free grammars (CFG)
- Understand that the languages specified by a CFG can be recognized by PDA
- Understand the subset of CFG such as LL, LR, SLR, LALR that are commonly used to specify a programming language
- Use LL, LR, SLR and LALR grammars to specify the syntax of a programming language
- Understand the characteristics of programming languages that cannot be define by CFG
- Understand the process of top-down parsing
- Understand the process of bottom-up parsing
- Use a parser generator to build a parser for a programming language
- Understand the role of the runtime stack and activations records in the execution of a
- program written in a programming language that supports recursion
- Create a simple compiler that can generate assembly or machine language code for programs written in a programming language that include common programming language constructs including arithmetic expressions, if statements, loops, procedure and function calls with parameters

CS 443 - Topics in Operating Systems

Catalog Description

An intermediate course in operating systems extending topics introduced in CS 441. Operating systems concepts are studied in depth. Typically students will study and modify an existing system. Prerequisite: CS 441. Offered Fall even-numbered years.

Prerequisites:

CS 441

Learning Objectives

Xinu[2]

Students shall be able to:

- Demonstrate, through several initial programming assignments, competence in using the C language, especially low-level features such as pointer manipulations.
- Understand the hardware and run-time environment of the computer on which the operating system runs.
- Read and understand the source code for fundamental components of the operating system:
  - CPU scheduling and context switching
  - process management (creation, termination, suspension, resumption)
  - process coordination
  - message-passing mechanisms (low level)
  - memory management (low level)
  - interrupt processing
  - real-time clock management
  - device-independent I/O management
  - device drivers
  - system initialization and bootstrapping
- Modify, rebuild, and test portions of the operating system to enhance functionality or improve performance.

[2] The specific learning objectives for this course will vary based on the specific operating systems examined. This document lists the SLO's for recent offerings.

CS 446 - Object Oriented Software Development

Catalog Description

Introduction to the concepts and principles of object-orientation (OO). Topics include detailed discussion on analysis and design of OO software systems, notations for OO analysis and design, and comparison of OO programming languages. Advanced topics on object-orientation such as OO testing and software reuse will be briefly discussed.

Prerequisites:

CS 340 and Junior standing.

Learning Objectives

Students shall be able to:

- Learn all UML models – use case diagrams, class and object diagrams, state diagrams, sequence diagrams, collaboration diagrams and packaging diagrams.
- Develop UML models (all diagrams) for a given software project, analyze them and use them for developing code. Implement and demonstrate the project.
- Learn about OO software life cycle, OO software testing and advanced OO technologies.

CS 449 - Advances in Software Engineering

Catalog Description

Introduces advanced topics in software engineering. Topics include prototyping models, risk analysis, component-oriented software development, software architectures, software reuse, software metrics and quality analysis.

Prerequisites:

CS 341 and Junior standing.

Learning Objectives

Students shall be able to:

CS 451 - User Interface Design

Catalog Description

This course focuses on the design and implementation of user interfaces. The topics include characteristics of user interfaces, user profiles, user interface design principles, methods and tools for user interface development, evolution of user interfaces, evaluation of user interfaces, and case studies.

Prerequisites:

CS 340 and Junior standing.

Learning Objectives

Students shall be able to:

- Learn about principles of graphical user-interface (GUI) design that include consistency of GUI elements, user-centered design, tools and techniques for GUI design, and usability testing.
- Apply GUI design principles to several problems and critically compare the features for various applications including data-oriented applications, temporal applications, applications that require graphics and web-based software products.
- Choose appropriate GUI development techniques for a software product.
- Learn about advanced GUI technologies.

CS 452 - Artificial Intelligence and Pattern Recognition

Catalog Description

An introduction to the fundamental principles of artificial intelligence. Topics include the biological basis for intelligence, classification of object descriptions and pattern recognition, search strategies and game trees, natural language processing, automatic theorem proving, programming for artificial intelligence and knowledge-based systems. Projects include writing a substantial artificial intelligence application program.

Prerequisites:

CS 340.

Learning Objectives

Students shall be able to:

- Discuss the fundamental questions of AI related to the characteristics of intelligence and how it is observed.
- Discuss the distinction between classification and recognition problems (supervised and non-supervised).
- Explain and solve problems related to pattern recognition, clustering and learning structure from samples.
- Explain and solve problems related to neural net structure and function.
- Explain and solve problems related to searching state spaces including the strategies of breadth-first, depth-first, heuristic, backtracking and pruning.
- Explain and solve problems related to searching and planning.
- Explain and solve problems related to rule-based systems.
- Explain and solve problems related to knowledge representation.
- Explain the difference between planning and learning approaches to AI problem solving.
- Explain the difference between on-line and off-line learning methods.
- Implement learning techniques for AI applications.
- Use declarative programming techniques to implement AI algorithms.

CS 453 - Introduction to Theory of Computation

Catalog Description

An introduction to the theoretical aspects of computation. The capabilities and limits of several computation models are considered including: partial recursive functions, Turing machines, finite state automata and formal languages. The implications of Church's thesis and unsolvable problems such as the halting problem are discussed.

Prerequisites:

CS 340 and Junior standing.

Learning Objectives

Students shall be able to:

- Explain and solve problems related to propositional logic, first-order predicate logic, elementary set theory (e.g. relations, functions, equivalence relations, partitions) and induction on computational structures.
- Explain and solve problems related to finite state machines, regular expressions and regular languages including the algorithms for transforming between these concepts and the pumping lemma.
- Explain and solve problems related to pushdown automata and context free languages including the algorithms for transforming between these concepts.
- Explain and solve problems related to Turing machines and partial recursive functions.
- Explain the significance of Church's Thesis.
- Explain and solve problems related to algorithmic complexity in time and space including transformation between problem types.
- Explain and solve problems related to the P and NP complexity classes.

CS 454 - Digital Image Processing

Catalog Description

This course introduces the fundamentals of digital image processing techniques with an emphasis on the design and implementation of image processing algorithms. Topics include: color models, point-processing techniques, convolution, Fourier domain processing, the discrete cosine transform, image compression methodologies, image restoration and enhancement, sampling and image display.

Prerequisites:

CS 340.

Learning Objectives

Students shall be able to:

- Describe the central elements of the human visual system including: rods; cones; cornea; pupil.
- Describe human perception including effects such as: simultaneous contrast; mach banding; brightness adaptation.
- Define and explain the following concepts:
    - resolution
    - color depth
    - color model (RGB, HSV, YIQ, CMY, CMYK)
    - aspect ratio
    - dot pitch
    - histogram
- Define and implement spatial domain processing operations including: gamma correction, scaling, histogram equalization, high/low/band pass, false coloring and blending.
- Construct the DFT and DCT coefficients of a small one-dimensional sequence by hand.
- Implement frequency domain processing operations including: high/low/band pass filtering.
- Encode and decode images that have been encoded as: run-lengths, delta-modulated, and quad-trees.
- Implement dithering using: random dithering, patterning, and error diffusion including Floyd-Steinberg dithering.
- Use and develop affine transformation matrices.
- Implement transformations for image scaling, rotation, shearing.
- Define connectedness and connected component.
- Define erosion, dilation, opening, and closing.
- Represent a component using run-length encoding and chain coding.
- Characterize a component using both statistical and geometric features including: first and second moments of inertia; circularity; length; perimeter; compactness; eccentricity

CS 455 - Fundamentals of Information Security

Catalog Description

This course presents the fundamental concepts of information security. Basic policies, techniques and tools for maintaining the security of host computers, information networks and computer software are presented. Elementary cryptography is explored with special attention to applications in data encryption, hashing and digital signatures. Fundamental security management procedures also are introduced, as are the legal and ethical issues associated with computer security. Students will be expected to apply the knowledge gained to construct security policies and practice security in the form of access privileges, firewalls, vulnerability scanners and intrusion detection tools.

Prerequisites:

CS 220 and Junior standing.

Learning Objectives

- **Security Concepts**
  - Students are able to define the concepts of confidentiality, availability and integrity as they relate to information security.
  - Students are aware of the relationships and tradeoffs among confidentiality, availability and integrity.
  - Students are able to define the concepts of authentication, non-repudiation, access control and privacy.
  - Students are aware that security applies to hardware, software and data
  - Students can define the following basic information security terms: threats and vulnerabilities, risk analysis, cost benefit analysis, attacks, exploits, controls
  - Students can categorize different strategies in the following methods of defense detection, recovery, deflection, deterrence, prevention
  - Students can define and apply the Principle of Weakest Link
  - Students are aware of the essential role of trust in any security policy or practice.
  - Students can explain the relationship between trust and information assurance
  - Students can explain the concepts of sophisticated attackers and script kiddies and their differences.
- **Security Management**
  - Students can explain the difference between security policy and practice.
  - Students can construct a comprehensive information security policy based on environmental characteristics.
  - Students can select appropriate practices for typical security policies.
  - Students can explain the role of risk analysis and risk mitigation and their importance in risk management.
  - Students can perform a basic risk analysis.
  - Students can perform basic security planning, including physical security and technological security.

- Students are aware of common social engineering attacks and can prescribe associated mitigation techniques.

- **Encryption**
  - Students can explain the different cryptographic techniques required to preserve integrity, confidentiality and authenticity.
  - Students can define encryption, decryption, plain text, cipher text, and can explain their role in cryptography.
  - Students can explain and identify keys, when used in cryptographic ciphers.
  - Students can explain and identify the difference between a stream cipher and a block cipher.
  - Students can define cryptanalysis, apply simple cryptanalysis and explain its implication for the success of failure of cryptographic algorithms.
  - Students can define, identify and offer simple examples of substitution ciphers.
  - Students can define, identify and offer simple examples of transposition ciphers.
  - Students can define, identify and offer simple examples of product ciphers.
  - Students can design a one-time pad cipher and explain its usefulness and limitations.
  - Students can explain the Shanon characteristics of good ciphers.
  - Students can analyze the quality of cryptographic algorithms using confusion, diffusion and unicity distance as metrics.
  - Students are aware of the basics, especially the concept of rounds, in Fiestel ciphers.
  - Students are able to explain the advantages, disadvantages, and applications of symmetric encryption.
  - Students are aware of the applications and limitations of the DES algorithm.
  - Students are aware of the applications and advantages of the AES algorithm.
  - Students can explain the difference between symmetric and asymmetric (PKI) encryption, can identify proper uses for each, and identify well-known algorithms of each type.
  - Students can design the basic mechanisms required to form a public key encryption, as described in the Diffie Hellman PKI concept.
  - Students can explain the basics of the RSA algorithm.
  - Students can construct a cryptographic hashing algorithm.
  - Students can explain the proper situations for applying cryptographic hashing.
  - Students can explain the need for key exchange protocols.
  - Students can design security symmetric authentication protocols.
  - Students can design security asymmetric authentication protocols.
  - Students can explain the purpose and fundamental protocols for network certificates.
  - Students can design an algorithm for constructing and using digital signatures.
  - Students are aware of steganography.
- **Software Security**
  - Students can explain how boot sectors, buffer overflows and memory leaks can be vulnerable to security attacks.
  - Students can offer examples of common computer viruses and their impact.

- Students can utilize software to scan for the presence of and vulnerabilities to malicious code.
- Students explain the use of virus signatures in scanning.
- Students can define the concepts of honey pots and secure sandboxes and their relationship to computer security.
- Students can explain fundamental software engineering practices including peer evaluations, tiger teams, modularity, encapsulation configuration management and verification; and can explain the importance of such practices to software security.

- **Host-level Security**
  - Students can explain how access control lists can be used to implement security policies.
  - Students can configure a host computer using access control list mechanisms.
  - Students can define the Principle of least privilege and explain why this is relevant to host-level security.
  - Students can define the challenge-response concept and how it is applied in security practices.
  - Students can explain the utility of password protection, be aware of common password cracking utilities and the associated password vulnerabilities.
  - Students can provide a collection of best use policies for password protection.
  - Students can explain the concepts of smartcards and tokens and can identify their strengths and weaknesses in user authentication.
  - Students are aware of biometric devices for user authentication.
  - Students can identify the primary operating system objects that require protection.
  - Students can explain how physical, temporal, logical, and cryptographic separation all play a role in operating system security.
  - Students are aware of the importance of proper memory management for security.
  - Students can implement file protections in the Windows operating environment.
  - Students can implement file protections in the Unix operating environment.
  - Students can configure a commonly available host firewall.
  - Students can define the concept of a trusted system.
  - Students can explain the Bell-LaPadula and related models of security.
  - Students can explain the Chinese wall policy and its applications in security.
- **Network Security**
  - Students know the names and roles of the seven layers of the OSI model for network transmission.
  - Students understand the role of network ports in providing network services.
  - Students can explain the fundamental rules of IPv6.
  - Students can explain the difference for UDP and TCP, the associated network services and their different security vulnerabilities.
  - Students are aware of the threats and vulnerabilities of such services as telnet, rlogin, bind and SNMP.
  - Students can explain the role of routers in network packet transmission.
  - Students can perform simple router configuration.

- Students can configure routers for packet filtering via access control lists.
- Students can explain the differences and relative advantages of static, dynamic and stateful firewalls.
- Students can configure a network firewall using access control lists.
- Students can explain the need for both network and host firewalls.
- Students can explain the differences between router filters, firewalls and intrusion detection systems.
- Students can explain the capabilities and limitations of intrusion detection systems.
- Students can explain the difference between intrusion detection and intrusion prevention.
- Students can explain the concept of virtual private networks and can identify their benefits and limitations.
- Students can design the proper placement of network devices for proper security.
- Students can explain the utility of the DMZ in network security layout.
- Students can explain the concept of secure tunneling and be aware of its use in providing hardened services such as SSL and HTTPS.
- Students can explain packet sniffing and the information it provides.
- Students can perform port scanning for the purpose of security audits
- Students are aware of Web browser vulnerabilities.
- Students can explain the use, and potential for misuse, of Web application cookies.
- Students can explain the basics of SQL injection attacks.
- Students can utilize CERT and other trusted Internet services to access the latest information on vulnerabilities and fixes.

- **Legal & Ethical Issues**
  - Students can define the concepts of intellectual property, copyrights, and patents
  - Students can explain the fundamental issues of employer/employee rights and responsibilities
  - Students can construct common acceptable use policies
  - Students can identify typical computer crimes.
  - Students can broadly explain the impact of the most significant laws and court opinions applicable to information security.

CS 456 - Secure Software Development

Catalog Description

Traditionally, software engineering has viewed flaws as the inconsistency of software behavior with its functional requirements. Software security problems, however, can occur in software that contains no such flaws but is nonetheless susceptible to external attack. This course examines known reasons for software security vulnerabilities with an emphasis on best practices for their detection and mitigation, along with general principles for engineering software in ways that enhance security.

Prerequisites:

CS 340

Learning Objectives

Students shall be able to:

- define and use the following terms*: confidentiality, integrity, availability assets, threats, faults, vulnerabilities, exploits, attacks, mitigations*
- understand security life cycles
- understand the environments (O.S., language, APIs, dbms, Internet) in which information security is relevant
- understand access control, patching, reporting (CERT and Bugtraq), backup (recovery and forensics) and auditing
- define and use the term *encryption*
- understand what it means to
    - secure the weakest link,
    - practice defense in depth
    - fail securely
    - compartmentalize
    - mediate completely
    - keep it simple
    - utilize least privilege
    - minimize attack surface
    - use open design
    - program defensively (be reluctant to trust)
- understand common application vulnerabilities including
    - numeric overflow
    - buffer overflow & stack smashing
    - inadequate input validation
    - race conditions (TOCTOU vulnerabilities)
    - pointer subterfuge and arc injection
    - package scope and inner classes
    - poor randomness
    - insufficient authentication

- understand common system-related vulnerabilities including
    - permission vulnerabilities and privilege elevation
    - I/O flaws
    - symlink vulnerabilities
- understand concepts and strategies related to protection including
    - Sandboxes, wrappers and jails
    - Static and dynamic analysis tools
    - Security in specifications
- understand network-related vulnerabilities including
    - Web application architecture
    - cookies
    - HTTPS/SSL
    - URL validation
    - SQL injection
    - socket and RPC programming
- understand thread modeling, including
    - threat modeling team assembly
    - software application decomposition
    - threat identification
    - threat ranking
    - response determination
- understand
    - the STRIDE classification for threats
    - the DREAD method of threat ranking
    - the OCTAVE method
- understand testing for security, including
    - difference between security testing & functional testing
    - security code reviews
    - vulnerability ranking
    - attack surface determination
    - penetration testing and data mutation testing
    - open source vs. closed source
    - privacy reviews
- understand general aspects of security laws, regulations and policies including
    - HIPAA
    - Sarbanes Oxley
    - Gramm-Leach-Bliley

CS 464 - Advanced Database Management Systems

Catalog Description

Advanced topics in database management systems. Topics include the relational data model, relational calculus, embedded SQL programming, database application programming, indexing, systems software and storage structures for databases, concurrency control, crash recovery, database administration, parallel and distributed databases, object oriented databases.

Prerequisites:

CS 364 and Junior standing.

Learning Objectives

Students shall be able to:

CS 470 - Parallel and Distributed Computing

Catalog Description

A study of architectures, control software, and applications for parallel and distributed systems. A survey of parallel and distributed architectures including data flow machines, vector processors, shared memory multiprocessors, and message based multiprocessors. Software topics include process communication and synchronization, global state maintenance, negotiation, scheduling, data parallelism, control parallelism, and languages for parallel and distributed computing.

Prerequisites:

CS 370 and Junior standing.

Learning Objectives

Students shall be able to:

- Understand the basic architecture of shared memory parallel computers including multicore processors and GPUs.
- Understand the basic architecture of distributed memory parallel computers
- Understand the design of interconnection networks
- Understand how a network of computers can be used for parallel computing
- Understand shared memory parallel programming models
- Understand distributed memory parallel programming models
- Understand how GPUs can be used for parallel programming
- Understand synchronization methods for parallel algorithms
- Create parallel applications using a programming interface based on threads
- Create parallel applications using a programming interface (such a OpenMP) based on
- Create parallel applications using a programming interface based on message passing
- Create parallel applications using a programming interface based on remote procedure calls
- Understand the potential advantages and limitations of parallel computing
- Evaluate the granularity of the potential
- Be able to match a parallel algorithm with an appropriate parallel architecture
- Understand the challenge of testing parallel applications
- Be able to analyze parallel algorithms for potential speed-up
- Be familiar with the idea of "Grand Challenge" problems and why parallel computing seems essential to finding solutions to these problems.

CS 471 - Data Communications

Catalog Description

An introduction to data communications, including the electrical properties and software protocols. In addition to presentations of the concepts and techniques used for data communications, several currently used standards and communications networks will be examined.

Prerequisites:

CS 270, CS 340 and Junior standing.

Learning Objectives

Students shall be able to:

- Define the Open System Interconnection model (OSI) and identify the responsibilities assigned to each layer in the model
- Understand the relationship between the TCP/IP architecture and OSI model
- Understand the encoding of binary values in a communication network
- Understand the purpose of frame level protocols and explain how frame level algorithms such as Stop-and-wait, Go-back-n and Selective repeat work
- Understand the role of checksums and CRC codes in detecting transmission errors
- Understand basic of local area networks concepts CSMA/CD, token ring and wireless networks
- Understand packet switching and circuit switching and the advantages and disadvantages of each type of switching
- Understand the basics of the IP protocol
- Understand routing algorithms including intra-domain routing and inter-domain routing
- Understand the concepts of subnetting and supernetting
- Be able to design small routing tables using subnettting and supernetting
- Understand the concept of multicasting
- Understand the basics of the TCP protocol
- Understand the problem of congestion control and various solutions that have been proposed to solve the problem
- Understand the concept of remote procedure call (RPC)
- Develop network applications that use TCP, UDP or RPC
- Understand the problem of data representation in the context of the development of network applications.
- Understand the basic issues in network security
- Understand the role cryptography plays in network security

CS 480 - Survey of Computer Assisted Instructional Systems

Catalog Description

A survey of current trends in Computer Assisted Instruction (CAI). Development of instructional and curriculum materials suitable for computer applications. Use of a current authoring software package.

Prerequisites:

(CS 224 or CI 420) and Junior standing.

Learning Objectives

Students shall be able to: