CSCI-COMPUTER SCIENCE

CSCI 1110. Computer Science Principles
4 Credits (3+2P)
This course provides a broad and exciting introduction to the field of computer science and the impact that computation has today on every aspect of life. It focuses on exploring computing as a creative activity and investigates the key foundations of computing: abstraction, data, algorithms, and programming. It looks into how connectivity and the Internet have revolutionized computing and demonstrates the global impact that computing has achieved, and it reveals how a new student in computer science might become part of the computing future.
Prerequisite: MATH 1215 or higher.
Learning Outcomes
1. Identify and differentiate programming constructs like IF, FOR, and WHILE.
2. Convert numbers between Hexadecimal, Binary and Decimal.
3. Write pseudo code to manipulate a robot.
4. Use an ASCII table to translate HEX strings into characters.
5. Encrypt and Decrypt simple messages with a Caeser Cypher.

CSCI 1115G. Modern Computing in Practice
4 Credits (3+2P)
This course provides a survey of practical and theoretical foundations for how computers work and how they support fundamental organizational needs. The course covers broad aspects of the hardware, software, and mathematical basis of computers. Lab assignments provide hands-on applications to use simple basic software tools to write simple programs, build and edit websites, analyze data with spreadsheets, choose an office productivity suite, and demonstrate computer literacy to potential employers. May be repeated up to 4 credits.
Prerequisite: MATH 1130G or MATH 1215 or higher.
Learning Outcomes
1. Students will create simple python programs using conditional statements and loops.
2. Students will analyze data with spreadsheet formulas, charts, and tools.
3. Students will create and publish a personal website using website building tools.
4. Students will edit HTML and CSS to format a website manually.
5. Students will practice the skill of performing software QA and providing actionable feedback to developers.
6. Students will become aware of common cybersecurity risks.
7. Students will learn basic vocabulary and context for broad aspects of hardware, software, and computer science theory such as Security, Privacy, Cloud Computing, the Internet, the Web, Operating Systems, Discrete Math, and Information Systems.
8. Students will be exposed to various sub-fields of CS including artificial intelligence, security, data analytics, UX, web development, and QA testing.
9. Students will reason about the societal impacts of technology. 1
10. Students will incorporate their new knowledge and skills into their resume.

CSCI 1120. Introduction to Computer Animation
3 Credits (3)
Introductory course for learning to program with computer animation as well as learning basic concepts in computer science. Students create interactive animation projects such as computer games and learn to use software packages for creating animations in small virtual worlds using 3D models. Recommended for students considering a minor/major in computer science or simply interested in beginning computer animation or programming.

CSCI 1210. Computer Programming Fundamentals
3 Credits (2+2P)
This course is an introduction to problem-solving methods and algorithm development. Students will learn how to design, code, debug, and document programs. Students will explore basic programming concepts including variables, data types, operators and expressions. Students will learn about input/output mechanisms, including command prompt interaction, and reading and writing data to files. Students will be introduced to control structures such as branching, conditionals, iteration, and loops and arrays. They will also learn how to define and use functions/methods to structure code and improve code reuse. May be repeated up to 3 credits.
Prerequisite(s): MATH 1215 or higher.
Learning Outcomes
1. Demonstrate an understanding of procedural programming techniques by implementing programs which employ structured programming techniques.
2. Implement control flow structures in programs to execute statements in a specified order, repeat sequences of statements, and execute different statements based on conditions.
3. Apply modularization principles by defining and using functions/methods to structure code and improve code reuse and maintainability.
4. Write code utilizing data structures such as arrays, simple classes and objects, to provide useful access to, and operations on, data.
5. Input/output mechanisms to collect user input and display data, including implementing error handling mechanisms to handle invalid input and output operations. the concept of recursion and identify base case and inductive step.

CSCI 1220. Computer Programming Fundamentals: Python
3 Credits (3)
This course is an introduction to fundamental programming concepts, with a focus on problem-solving techniques and algorithm development using the Python programming language. Students will learn how to create basic scripts, work with data types and variables, use control structures, and build functions. The course is designed for students with little or no prior experience in programming and is intended to provide a foundation in programming that can be applied in a variety of fields.
Prerequisite: MATH 1215 or higher.
Learning Outcomes
1. Apply programming concepts to design and develop solutions for computational problems.
2. Utilize optimal flow-control strategy for solving a given problem.
3. Design and implement functions to support organization, decomposition and reusability of code.
4. Evaluate and select data structures to efficiently organize and store information for a given problem.
5. Demonstrate the concept of scope to control access to global, local, and class variables.
6. Create and use a simple class to demonstrate object-oriented programming principles.
7. Utilize basic file input/output with text-based files.

CSCI 1225. Python Programming II
3 Credits (3)

This course covers advanced Python programming, including classes, objects, and inheritance, embedded programming in domain applications, database interaction, and advanced data and text processing. The focus will be on preparing students to use Python in their own areas.
Prerequisite(s): CSCI 1220 or CSCI 4520.

CSCI 1235. R Programming I
3 Credits (3)
This course is an introduction to data processing in the R language, covering fundamental script configuration, data types and data collections, R control structures, and basic creation of graphs and data visualizations. This course will not focus on the statistical capabilities of R, though some basic statistical computations will be used.
Prerequisite(s): MATH 1220G.

CSCI 1240. C++ Programming I
3 Credits (2+2P)
This course is an introduction to problem-solving methods and algorithm development using C++. Students will learn how to design, code, debug, and document programs. Students will explore basic programming concepts including variables, data types, operators and expressions. They will also learn how to work with the C++ preprocessor directives and libraries. Students will learn about input/output mechanisms, including command prompt interaction, reading and writing data to files. Students will be introduced to control structures such as branching, conditionals, iteration, and loops and arrays. They will also learn how to define and use functions to structure code and improve code reuse.
Prerequisite: MATH 1215 or higher.
Learning Outcomes
1. Implement programs which employ structured programming techniques.
2. Implement control flow structures.
3. Apply modularization principles by defining and using functions/methods to structure code and improve code reuse and maintainability.
4. Write code utilizing data structures such as arrays, using pointers, and simple classes to provide useful access to, and operations on, data.
5. Use input/output mechanisms to collect user input and display data, including implementing error handling mechanisms to handle invalid input and output operations.

CSCI 1720. Computer Science I
4 Credits (3+2P)
Computational problem solving; problem analysis; implementation of algorithms using Java. Object-oriented concepts, arrays, searching, sorting, and recursion.
Prerequisite: (A C- or better in either MATH 1250G or (MATH 1430G or higher)) OR (A C- or better in MATH 1220G and a 1 or better in the CS Placement Test) OR (A C- or better in MATH 1220G and a C- or better in CSCI 1110).
Learning Outcomes
1. Develop algorithms to solve problems.
2. Implement algorithms using the fundamental programming features of sequence, selection, iteration, and recursion.
3. Apply an understanding of primitive and object data types.
4. Design and implement classes based on given attributes and behaviors.
5. Explain the fundamental concepts of object-oriented programming.

CSCI 2210. Object-Oriented Programming
4 Credits (3+2P)
This course is an introduction to object-oriented programming. Including: Classes and objects, and associated topics such as constructors, properties, and methods, inheritance, polymorphism, encapsulation, abstraction, exception handling and best practices. May be repeated up to 4 credits.
Prerequisite: At least a C- in CSCI 1720 or ENGR 140.
Learning Outcomes
1. Implement object-oriented designs based on project requirements.
2. Use encapsulation to write programs that are loosely coupled and easy to debug, maintain and modify.
3. Use inheritance to define simple class hierarchies that allow code to be reused by distinct subclasses.
4. Implement and reason about control flow in a program using polymorphism to solve common programming problems.

CSCI 2220. Introduction to Data Structures and Algorithms
4 Credits (3+2P)
Design, implement, and use fundamental abstract data types including linked lists, stacks, queues, and trees. Analyze the time and space complexity of algorithms, such as sorting.
Prerequisite: At least a C- in CSCI 1720, or placement.
Learning Outcomes
1. Implement basic data structures such as linked lists, stacks, queues, and trees in a high-level programming language.
2. Compare alternative implementations of data structures with respect to time and space complexity.
3. Explain the advantages and disadvantages of a variety of sorting algorithms.

CSCI 2230. Assembly Language and Machine Organization
4 Credits (3+2P)
Computer structure and system organization, instruction execution, memory addressing modes, hardware/software interface. Programming in assembly language. May be repeated up to 4 credits.
Prerequisite: At least a C- in CSCI 1720 or ENGR 140.
Learning Outcomes
1. Describe the architecture of a microcontroller, the interconnections between the components, and the major units inside the CPU.
2. Use signed and unsigned numbers, bitwise operations, branching instructions, and the corresponding flags in the status register.
3. Use immediate, direct, indirect addressing modes in assembly instructions.
4. Map high-level programming language features to assembly instructions, such as loops, conditionals, procedure calls, value and reference parameter passing, return values, and recursion.
5. Interface with input/output devices via instructions, memory addressing, or interrupts.
6. Design and implement an assembly language program.

CSCI 2310. Discrete Mathematics for Computer Science
4 Credits (3+2P)
Discrete mathematics required for Computer Science, including the basics of logic, number theory, methods of proof, sequences, mathematical induction, set theory, counting, and functions. Taught with CSCI 4560.
Prerequisite: At least C- in CSCI 1720.
Learning Outcomes
1. Use logic to specify precise meaning of statements, demonstrate the equivalence of statements, and test the validity of arguments.

2. Construct and recognize valid proofs using different techniques including the principle of mathematical induction.
3. Use summations, formulas for the sum of arithmetic and geometric sequences.
4. Explain and apply the concepts of sets and functions.
5. Apply counting principles to determine the number of various combinatorial configurations.

CSCI 2410. Practical Programming
2 Credits (1+1P)
A hands-on dive into practical programming skills development. Students will practice skills such as implementing algorithms that manipulate data in arrays and other data structures, implementing and using hashing-based data collections, using I/O in programs to access and create data, and object-oriented programming. Students will also focus on honing their use of tools such as command line, integrated development environments, debuggers, and profilers for software development. May be repeated up to 2 credits.

Learning Outcomes

1. Perform simple manipulation of arrays and other basic data structures.
2. Better utilize objects and object oriented programming.
3. Utilize different tools for building, debugging, and improving their programs.
4. Will be able to learn and use a new programming language quickly.
5. Use basic I/O capabilities in a variety of languages.
6. Use documentation to learn important features of a programming languages.
7. Write programs that solve interview-like problems.

CSCI 2996. Special Topics
1-3 Credits
Varies.

Learning Outcomes

1. Varies.

CSCI 3410. Introduction to Intelligent Agents Using Science Fiction
3 Credits (3)
This course uses science-fiction movies to introduce fundamental principles and techniques in agents and multi-agent systems. It is a gentle introduction to decision theory, machine learning, multi-agent systems, and ethics in agent-based systems.

Learning Outcomes

1. Use decision-theoretic models and algorithms to represent and solve simple planning and reasoning problems under uncertainty.
2. Use Markov Decision Processes to model and solve planning and reinforcement learning problems.
3. Use game-theoretic models and algorithms to represent and solve simple game- theoretic problems.
4. Understand the tradeoffs between the different agent models.
5. Understand the challenges for ensuring that AI agents are safe as they play an increasingly large role in modern society.

CSCI 3710. Software Development
4 Credits (3+2P)
Software specification, design, testing, maintenance, documentation; informal proof methods; team implementation of a large project. Taught with CSCI 4575.
Prerequisite: At least a C- in CSCI 2710 and CSCI 2220.

Learning Outcomes

1. Understand and explain the activities and structure of different styles of software development processes, including waterfall, (spiral,) iterative, and agile methodologies.
2. Apply requirements knowledge and techniques to create functional and non-functional requirements for a software system.
3. Apply high and low level design ideas to create an object-oriented design of a software system.
4. Use good design and programming ideas to implement individual and team software systems in compiled OOP languages.
5. Apply white and black box testing techniques and tools to individual and team software development.
6. Use UML class diagrams (and sequence diagrams) to capture aspects of system design and/or requirements (domain).
7. Use practical software development tools, including version control systems, automated build tools, and testing tools.

CSCI 3720. Data Structures and Algorithms
4 Credits (3+2P)
Introduction to efficient data structure and algorithm design. Order notation and asymptotic run-time of algorithms. Recurrence relations and solutions. Abstract data type dynamic set and data structures based on trees. Classic algorithm design paradigms: divide-and-conquer, dynamic programming, greedy algorithms. Taught with CSCI 5110. May be repeated up to 4 credits.
Prerequisite: At least a C- in CSCI 2220 and CSCI 2310.

Learning Outcomes

1. Analyze the growth of functions via asymptotic notation.
2. Evaluate the asymptotic running time of a given algorithm.
3. Solve recurrence relations of the kinds encountered in algorithm analysis.
4. Design algorithms using the divide-and-conquer technique.
5. Design algorithms using the greedy technique.
6. Design algorithms using the dynamic-programming technique.
7. Use and analyze data structures based on trees.
8. Analyze the design, correctness, and time complexity of basic graph algorithms.

CSCI 3730. Compilers and Automata Theory
4 Credits (3+2P)
Methods, principles, and tools for programming language processor design; basics of formal language theory (finite automata, regular expressions, context-free grammars); development of compiler components. Taught with CSCI 4580.
Prerequisite: At least a C- in CSCI 2210, CSCI 2220, and CSCI 2230.

Learning Outcomes

1. Understand the language theory concepts of regular languages, context free languages, regular expressions, context free grammars, and formal language hierarchy.
2. Use Thompson's construction to convert from regular expression to NFA, and subset construction to convert from NFA to DFA.
3. Apply recursive descent parsing in programming a parser of a small grammar.
4. Understand the ideas in LL and LR parsing of context-free language classes.
5. Understand and use table-driven top-down (LL(1)) and bottom up (SLR) parsing to parse a sentence.

CSCI 3790. Algorithm Design & Implementation
3 Credits (3)
Introduction to efficient data structure and algorithm design. Basic graph algorithms. Balanced search trees. Classic algorithm design paradigms: divide-and-conquer, greedy scheme, and dynamic programming. Taught with CSCI 4590.
Prerequisite: At least a C- in CSCI 2220, or consent of instructor.
Learning Outcomes
1. Be able to use and implement sorting algorithms.
2. Be able to design and implement graph algorithms.
3. Be able to design and implement algorithms using the divide-and-conquer technique.
4. Be able to design and implement algorithms using the greedy technique.
5. Be able to design and implement algorithms using the dynamic programming technique.
6. Be able to use and implement balanced search trees.
7. Be able to use and implement hashing techniques.
8. Be able to perform the run time analysis of basic algorithms using Big O notation.

CSCI 3997. Independent Study
1-6 Credits (1-6)
Faculty supervised investigation, to culminate in a written report. May be repeated up to 6 credits.
Learning Outcomes
1. Varies.

CSCI 4105. Programming Language Structure I
3 Credits (3)
Syntax, semantics, implementation, and application of programming languages; abstract data types; concurrency. Not for Computer Science graduate students.
Prerequisite: At least a C- in CSCI 3730 and CSCI 3710.
Learning Outcomes
1. Improve the background for choosing appropriate programming languages for certain classes of programming problems.
2. Increase the ability to learn new programming languages.
3. Critically evaluate what paradigm and language are best suited for a new problem.
4. Demonstrate the use of the primary segments for a running program.
5. Apply the principles of functional programming.
6. Apply the principles of logic programming.
7. Program a simple parallel program with threads.
8. Program in at least five different programming languages.
9. Program in C to demonstrate architecture details.

CSCI 4110. Computing Ethics and Social Implications of Computing
1 Credit (1)
An overview of ethics for computing majors includes: history of computing, intellectual property, privacy, ethical frameworks, professional ethical responsibilities, and risks of computer-based systems.
Prerequisite: At least a C- in CSCI 3710.
Learning Outcomes
1. Understand the fundamental technologies and operation of the web.
2. Design and develop responsive interactive web sites.
3. Deploy web applications on Cloud Computing Platforms.
4. Leverage modern tools and packages to develop full stack web applications.
5. Be fluent in the application of emerging web technologies like browser extensions, WebSockets, and WebRTC.
6. Use existing materials and references on the web to learn new skills.

CSCI 4120. Operating Systems I
3 Credits (3)
Operating system principles and structures, and interactions with architectures. Not for Computer Science graduate students.
Prerequisite: At least a C- in CSCI 2230, CSCI 3710, and CSCI 3720.
Learning Outcomes
1. Explain OS control and management of hardware resources.
2. Explain OS management and execution of processes.
3. Explain OS control and management of real and virtual memory.
4. Explain classical concurrency issues and their solutions.
5. Analyze and implement threads.
6. Analyze OS interaction with networks and architecture.

CSCI 4130. Linux System Administration
3 Credits (3)
Basic system administration for Linux environments. Topics include user managements, file systems, security, backups, system monitoring, kernel configuration and other relevant aspects of system administration. Not for Computer Science graduate students
Learning Outcomes
1. Understand the architecture of a Linux system and software licensing (Linux's principles and philosophy).
2. Use common Linux commands for system installs, upgrades, and maintenance.
3. Use a Linux Command Line Interface for navigation and understanding the file system structure.
4. Recognize processes, automation and scripting tasks.
5. Utilize basic system security and managing file systems, user accounts, and file and folder ownership and permissions.
6. Manage and troubleshoot network configurations.
7. Manage and understand Domain Name Servers, Network File Systems, Web servers, and other common Linux applications.

CSCI 4140. Database Management Systems I
3 Credits (3)
Database design and implementation; models of database management systems; privacy, security, protection, recovery. Not for Computer Science graduate students. Taught with CSCI 5140.
Prerequisite: At least a C- in CSCI 2220 and CSCI 2310.
Learning Outcomes
1. Utilize the basic concepts of relational database model.
2. Utilize database query languages (e.g. SQL).
3. Identify data integrity and security requirements.
4. Analyze, capture, and model user requirements for building database systems using conceptual models.
5. Design and normalize relational schemas.
6. Apply application development methods to implement a database system.

CSCI 4215. Parallel Programming
3 Credits (3)
Programming of shared memory and distributed memory machines; tools and languages for parallel programming; techniques for parallel programming; parallel programming environments. Not for Computer Science graduate students. Taught with CSCI 5215.

Prerequisite: At least a C- in CSCI 3730 or consent of instructor.

Learning Outcomes

1. Describe existing parallel architectures including shared memory versus distributed memory platforms.
2. Apply basic techniques for organizing parallel computations.
3. Apply basic techniques for performance measurement and theoretical limitations of parallelism.
4. Explain alternative parallel techniques and hardware.
5. Perform performance Analysis of different parallel programming techniques.
6. Program shared memory machines using threads, processes, and the OpenMP library.
7. Program using a message passing paradigm and obtain working knowledge of the Message Passing Interface (MPI).

CSCI 4220. Cloud and Edge Computing

3 Credits (3)

The course presents a top-down view of cloud computing, from applications and administration to programming and infrastructure. Its main focus is on the concepts of networking and parallel programming for cloud computing and large scale distributed systems which form the cloud infrastructure. The topics include: overview of cloud computing, cloud systems, parallel processing in the cloud, distributed storage systems, virtualization, security in the cloud, and multicore operating systems. Students will study state-of-the-art approaches to cloud computing followed by large cloud corporations, namely Google, Amazon, Microsoft, and Yahoo. Students will also apply what they learn through project developments using Amazon Web Services. Not for graduate Computer Science majors. Taught with: CSCI 5220.

Prerequisite: At least a C- in CSCI 3720; background in CSCI 4245/ CSCI 5245 is preferred or consent of instructor.

Learning Outcomes

1. The emphasis of the course will be on the understanding the concepts and the engineering trade-offs involved in the design of cloud computing systems.
2. Cloud deployment models, cloud service models (software-as-a-service, infrastructure- as-a-service, protocol-as-a-service), cloud architecture, cloud-edge security, service level agreements, and load balancing in cloud and datacenters.
3. Learn about cloud computing, especially what are their fundamental components, how these components interact, and how the technology is evolving for the future (edge computing, cloudlets, mobile edge computing, etc.).

CSCI 4225. Introduction to Cryptography

3 Credits (3)

The course covers basic cryptographic primitives, such as symmetric, public-key ciphers, digital signature schemes, and hash functions, and their mathematical underpinnings. Course helps students understand basic notions of security in a cryptographic sense: chosen plaintext and chosen ciphertext attacks, games, and reductions. Course also covers computational number theory relevant to cryptography. Consent of Instructor required. Taught with: CSCI 5225.

Prerequisite: CSCI 2310 (or equivalent) with a C or better.

Learning Outcomes

1. Describe basic cryptographic primitives, including symmetric ciphers, asymmetric ciphers, digital signatures, message authentication codes, and hash functions.
2. Understand the mathematical, fundamental underpinnings of cryptography, and how to reason about the security of crypto primitives: indistinguishability (IND) properties of ciphertexts, CPA/ CCA games, and reductions to fundamental math assumptions.
3. Be able to discuss number theory/algebra underpinning the design of cryptographic primitives, in some depth.

CSCI 4235. Cellular Networks and Mobile Computing

3 Credits (3)

This course will offer a solid introduction to major global wireless standards and comparisons of the different wireless technologies and their applications and examine each technology and how to utilize several different systems for the best results. A basic understanding of Computer Networks is preferable as a course prerequisite.

Learning Outcomes

1. Understand user associations and routing in a cellular/mobile network.
2. Develop insight into interaction of elements within the cellular/mobile core.
3. Understand the concept of end-to-end delivery of a packet and/or signal.
4. Develop an understanding of what happens with the hand-off at each step along the communications path.
5. Be able to explain differences in core architecture between different generations of cellular and mobile network technologies.

CSCI 4240. Software Reverse Engineering

3 Credits (3)

This class provides students with fundamental experience in software reverse engineering with a focus on malware reverse engineering. Students will learn operational security for safely analyzing untrusted code in a sandbox environment. Students will learn control flow integrity attacks, binary control flow analysis, and how to analyze live program behavior. Taught together with CSCI 5240.

Learning Outcomes

1. Students will learn how malware behaves, spreads, and is controlled.
2. Students will learn how to safely analyze malware in controlled environments.
3. Students will learn how malware seeks to hide in systems.
4. Students will learn to perform static analysis of binaries using simple tools.
5. Students will learn how malware obfuscates itself to avoid analysis, including using crypto packers, polymorphism, and sandbox detection.
6. Students will learn to perform decompilation and control-flow analysis of binaries using Ghidra.
7. Students will learn to dynamically analyze malware in a sandbox environment while observing network traffic, resource consumption, and system calls.
8. Students will learn to detect malware running with operating-system level permissions (rootkits).
9. Students will learn memory forensics techniques to detect malware hidden within benign processes.

CSCI 4245. Computer Networks I

3 Credits (3)

Fundamental concepts of computer communication networks: layered network architecture, network components, protocol stack and service. Example of application, transport, network and data link layers, protocols primarily drawn from the Internet (TCP, UDP, and IP) protocol multimedia networks; network management and security. Not for Computer Science graduate students. Taught with CSCI 5245.

Prerequisite: At least a C- in CSCI 2220 and CSCI 2230.

Learning Outcomes

1. Understand how to break down the Internet into layers of the OSI model and how each layer of abstraction manages complexity.
2. Understand how data is encoded at the physical layer over copper, fiber, and RF, and the importance of framing and collision avoidance.
3. Understand the concept of packet switching networks, switch fabrics, the ARP, the DHCP, OSPF, and NAT.
4. Understand Internet organization and governance including IANA, ASes, IXPs, ISPs, CAs, and the BGP.
5. Understand the TCP/IP paradigm, including flavors of self-clocking, congestion control, the need for ports, and the end-to-end argument.
6. Understand common application-layer protocols including HTTP(S), FTP, SMTP, etc.
7. Understand the security and privacy guarantees and non-guarantees of TLS, and how they are achieved.
8. Understand the inherent consensus challenges of networked computing, and classical solutions such as the NTP and Lamport Clocks.
9. Write networking program in C that implements an application-layer protocol, directly using system calls and managing memory.

CSCI 4250. Human-Centered Computing

3 Credits (3)

Covers iterative, human-centered interface design, including prototyping and evaluation. Basics of graphic design and visualization. Not for Computer Science graduate students. Taught with CSCI 5250.

Prerequisite: At least C- in CSCI 3710.

Learning Outcomes

1. Describe, analyze, and/or critique a device interface using a design vocabulary.
2. Enact a human-centered process of interaction design: gather data; develop a data-driven design; iterate design through testing; and evaluate results.
3. Conduct human-computer interaction research by proposing, developing, and conducting experiments; analyzing data; and developing synthesized results.
4. Communicate design and evaluation with presentations, demos, and reports.
5. Implement a variety of interaction techniques.

CSCI 4255. Digital Game Design

3 Credits (3)

An introduction to digital game design. Topics include design, development, and playtesting of games. The course is structured to use team-based learning. Not for Computer Science graduate students. Taught with CSCI 5255.

Prerequisite/Corequisite: CSCI 3710.

Learning Outcomes

1. Describe, analyze, and/or critique games with a consistent vocabulary.
2. Design, develop, and playtest games.
3. Understand the formal systems of games.
4. Communicate game designs through demonstrations and presentations.

CSCI 4265. Modern Web Technologies

3 Credits (3)

In this course, we will take a full-stack approach to modern web application design. We will start with the fundamentals including HTML5, CSS3, Javascript, JSON, and the underlying networking concepts and protocols driving the modern web. We will then move on to more advanced topics including javascript backend development with Node.js, NoSQL database design with MongoDB, cloud computing, and re- sponsive web design. Finally, we cover advanced topics including the design and im- plementation of browser extensions and real-time web technologies like WebRTC and WebSockets. Consent of Instructor required. Taught with: CSCI 5265.

Learning Outcomes

1. Understand the fundamental technologies and operation of the web.
2. Design and develop responsive interactive web sites.
3. Deploy web applications on Cloud Computing Platforms.
4. Leverage modern tools and packages to develop full stack web applications.
5. Be fluent in the application of emerging web technologies like browser extensions, WebSockets, and WebRTC.
6. Use existing materials and references on the web to learn new skills.

CSCI 4270. Principles of Virtual Reality

3 Credits (3)

This course is an introduction to building systems and doing research in / on virtual reality. We cover system design, development, and evaluation, with an emphasis on recent research in the space. We cover a range of methods, qualitative and quantitative, in order to develop insights into effective VR designs. Students in this class will develop a foundation in VR development; learn about current topics in VR; and design, develop, evaluate, and report on a VR system.

Prerequisite: CSCI 4250.

Learning Outcomes

1. Design and develop systems in virtual reality.
2. Understand the variety of development techniques in VR.
3. Understand the state-of-the-art in VR systems.
4. Communicate understanding of people, designs, and evaluations through presentations, demos, and/or reports.

CSCI 4310. Bioinformatics Programming

3 Credits (3)

Computer programming to analyze high-throughput molecular biology data including genomic sequences, bulk and single-cell transcriptome, epigenome, and other omics data. Quality control, library size normalization, confounding effect removal, clustering, statistical modeling, trajectory inference, and visualization. Taught with CSCI 5310.

Learning Outcomes

1. Write R scripts and functions to manipulate biological sequences, genome annotation, and gene expression data.
2. Perform high-throughput data analysis with established R packages.
3. Detect differential gene expression on RNA sequencing data.
4. Perform single-cell RNA sequencing data analysis (quality control, library size normalization, confounding effect removal, modeling).
5. Assess statistical significance of analytical results.
6. Create automatic data analysis pipeline to link multiple software packages.

CSCI 4410. Computer Graphics I

3 Credits (3)

Languages, programming, devices, and data structures for representation and interactive display of complex objects. Not for Computer Science graduate students. Taught with CSCI 5405.

Prerequisite: At least C- in CSCI 3730 or CSCI 3710.

Learning Outcomes

1. Techniques used in three-dimensional graphics.
2. Computer Graphics lightning and shading.
3. Client-server graphics using WebGL.
4. Geometric and Solid modeling.
5. Computer Graphics implementation algorithms.

CSCI 4415. Introduction to Data Mining
3 Credits (3)
Techniques for exploring large data sets and discovering patterns in them. Data mining concepts, metrics to measure its effectiveness. Methods in classification, clustering, frequent pattern analysis. Selected topics from current advances in data mining. Taught with CSCI 5415.
Prerequisite: At least a C- in CSCI 220 and CSCI 2310.
Learning Outcomes

1. Explain and recognize different data mining tasks such as data pre-processing, visualization, classification, regression, clustering, association rules, and anomaly detection.
2. Apply classical data mining / machine learning algorithms for classification, clustering, association rules, and anomaly detection.
3. Evaluate and compare the performance of different data mining / machine learning algorithms.
4. Utilize data mining algorithms to analyze data in real applications using a data mining tool.

CSCI 4430. Graph Data Mining
3 Credits (3)
The course covers graph terminology, representation, and techniques to extract patterns from large graph data. The topics include random and scale-free graph generation, link analysis (PageRank), graph representation learning, graph neural networks, deep graph generation, community detection, frequent subgraph mining, graph classification, influence maximization, and knowledge graph mining.
Prerequisite: At least a C- in CSCI 2220 or CSCI 1220, and CSCI 2310, or consent of instructor.
Learning Outcomes

1. Have significant familiarity with different state-of-the-art theories and practices of graph data mining.
2. Graph representation and graph querying using graph manipulating toolbox/library.
3. Use random and scale-free graph models to generate graphs and visualize complex network properties.
4. Apply algorithms such as PageRank, spectral clustering, and non-negative matrix factorization.
5. Implement graph representation learning algorithms and graph neural networks.
6. Understand much of the current literature on the topic, review papers, extend their knowledge through further study, and present findings of the papers.

CSCI 4435. Text Mining and Natural Language Processing
3 Credits (3)
This course is an introduction to text mining and natural language processing (NLP). It covers NLP techniques for extracting insights from unstructured text data. Topics include text classification, semantic textual similarity, topic modeling, sentiment analysis, text summarization, text generation, and machine translation.
Prerequisite: At least a C- in CSCI 2220 and CSCI 2310.
Learning Outcomes

1. Describe and apply techniques for text processing, text representation, and text modeling.
2. Describe and apply machine learning algorithms for text mining and NLP tasks such as text classification, semantic textual similarity, topic modeling, sentiment analysis, text summarization, text generation, and machine translation.
3. Utilize Python and popular libraries for implementing NLP-based applications.
4. Evaluate the performance of text mining and NLP algorithms.

CSCI 4440. Generative Artificial Intelligence
3 Credits (3)
Covers the theory and applications of generative artificial intelligence. Concentration will be on specific topics such as large language models, adversarial neural networks, neural symbolic computing, and inductive logic programming. May be repeated up to 3 credits.
Learning Outcomes

1. Understand the theoretical foundation of generative AI tools.
2. Understand the strengths and weaknesses of generative AI tools and identify appropriate tools for a given application.
3. Utilize advanced generative AI tools such as multi-modal LLMs for problem-solving and developing practical applications.
4. Understand the ethical consequence of using generative AI tools.

CSCI 4510. C++ Programming
3 Credits (3)
Programming in the C++ language. Taught with CSCI 1240. More advanced than CSCI 1240. Recommended for nonmajors only. Not for Computer Science undergraduate students.
Learning Outcomes

1. Use various data types and the corresponding operations.
2. Write C++ programs that contain expressions, program control, functions, arrays, and input/output.
3. Explain basic object-oriented programming concepts.
4. Demonstrate proficiency in using classes, inheritance, pointers, streams, and recursion.

CSCI 4520. Python Programming I
3 Credits (3)
This course is an introduction to programming in the Python language, covering fundamental scripts, data types and variables, functions, and simple object creation and usage. The focus will be on preparing students to use Python in their own areas. No prior programming experience is required. Taught with CSCI 1220. More advanced than CSCI 1220.
Learning Outcomes

1. Develop an algorithm to solve a problem.
2. Demonstrate the ability to use Python data types: int, float, strings, and lists; and the built-in functions associated with those data types.
3. Edit and debug programs using the Spyder IDE for Python.
4. Implement algorithms using the Python features of assignment, input, output, branches, loops, and functions.
5. Explain the fundamental concepts of object-oriented programming with Python.
6. Design and implement Python classes based on given attributes and behaviors.
7. Work with existing Python modules such as math, random, and os.
8. Write Python programs that input data from files and store results in files.

CSCI 4525. Python Programming II
3 Credits (3)
This course covers advanced Python programming, including classes, objects, and inheritance, embedded programming in domain applications, database interaction, and advanced data and text processing. The focus will be on preparing students to use Python in their own areas. For graduate students only. Has more advanced work than CSCI 1225, and does not count towards Computer Science major requirements. Computer Science students are excluded from taking this course.
Prerequisite(s): CSCI 1220 or CSCI 4520.

CSCI 4540. Computer Science I Transition
3 Credits (3)
Computational problem solving; problem analysis; implementation of algorithms. Recursive structures and algorithms. For Computer Science graduate students only; cannot be used to meet a Computer Science student's program of study. Taught with CSCI 1720.
Learning Outcomes
1. Develop algorithms to solve problems.
2. Implement algorithms using the fundamental programming features of sequence, selection, iteration, and recursion.
3. Apply an understanding of primitive and object data types.
4. Design and implement classes based on given attributes and behaviors.
5. Explain the fundamental concepts of object-oriented programming.

CSCI 4545. Object Oriented Programming Transition
3 Credits (3)
Introduction to problem analysis and problem solving in the object-oriented paradigm. Practical introduction to implementing solutions in the C++ language. Hands-on experience with useful development tools. Cannot be used in a Computer Science student's program of study. Taught with CSCI 2210.
Prerequisite: At least a C- in CSCI 1720 or CSCI 4540 or consent of instructor.
Learning Outcomes
1. Develop an algorithm to solve a problem.
2. Implement algorithms using the C and C++ languages including imperative and object-oriented language features.
3. Demonstrate a noticeable increase in understanding of problem analysis and program design.
4. Demonstrate proficiency in using control structures including if statements (single selection), switch (multiple selection), and loops (repetition).
5. Demonstrate proficiency in using arrays and functions.
6. Create UML class and relationship diagrams.
7. Design a class to model a real-world person, place, thing, or event.
8. Use editing and debugging software to create, debug, and test C and C++ programs.
9. Understand the basic terminology used in object-oriented programming. 1
10. Create a make file to build an executable from a set of C or C++ source files.

CSCI 4550. Introduction to Data Structures Transition
3 Credits (3)
Design, implementation, use of fundamental abstract data types and their algorithms: lists, stacks, queues, deques, trees; imperative and declarative programming. Internal sorting; time and space efficiency of algorithms. Cannot be used in a C S student's program of study. Consent of Instructor required. Taught with CSCI 2220.
Prerequisite: At least a C- in CSCI 1720 or CSCI 4540 or consent of instructor.
Learning Outcomes
1. Be able to implement and use lists.
2. Be able to implement and use stacks.
3. Be able to implement and use queues.
4. Be able to implement and use trees.
5. Be able to perform the run time analysis of basic algorithms using Big O notation.
6. Be able to implement, use, and analyze searching algorithms.
7. Be able to solve a problem recursively.
8. Take a problem statement from a user and convert it into a Java program that fulfills the user's needs.
9. Create object oriented Java classes that effectively separate and hide implementation details from client applications.

CSCI 4555. Machine Programming and Organization Transition
3 Credits (3)
Computer structure, instruction execution, addressing techniques; programming in machine and assembly languages. Cannot be used in a Computer Science student's program of study. Taught with CSCI 2230.
Prerequisite: At least a C- in CSCI 1720 or CSCI 4540 or consent of instructor.
Learning Outcomes
1. Describe the architecture of a microcontroller, the interconnections between the components, and the basic units inside the CPU.
2. Use signed and unsigned numbers, the associated branching instructions, and the corresponding flags in the status register.
3. Explain immediate, direct, indirect addressing modes, their opcode and operands, and their utilities.
4. Map high-level programming language features to assembly instructions, including loops, conditionals, procedure calls, value and reference parameter passing, return values, and recursion.
5. Interface with I/O devices including LED and sensors via digital input and output, and analog-to-digital conversion.
6. Program timers/counters and interrupts to control real-time applications.
7. Design an assembly program.

CSCI 4560. Discrete Math for Computer Science Transition
3 Credits (3)
Logical connectives, sets, functions, relations, graphics, trees, proofs, induction, and application to computer science. For Computer Science graduate students only. Cannot be used in a Computer Science student's program of study. Taught with CSCI 2310.
Prerequisite: At least a C- in CSCI 1720 or CSCI 4540 or consent of instructor.
Learning Outcomes
1. Use logic to specify precise meaning of statements, demonstrate the equivalence of statements, and test the validity of arguments.
2. Construct and recognize valid proofs using different techniques including the principle of mathematical induction.
3. Use summations, formulas for the sum of arithmetic and geometric sequences.
4. Explain and apply the concepts of sets and functions.

5. Apply counting principles to determine the number of various combinatorial configurations.

CSCI 4575. Software Development Transition
3 Credits (3)
Software specification, design, testing, maintenance, documentation; informal proof methods; team implementation of a large project. For Computer Science graduate students only. Cannot be used in a Computer Science student's program of study. Taught with CSCI 3710.
Prerequisite(s): At least a C- in CSCI 271 or CSCI 4545, in CSCI 2220 or CSCI 4550, or consent of instructor.
Learning Outcomes
1. Understand and explain the activites and structure of different styles of software development processes, including waterfall, (spiral,) iterative, and agile methodologies.
2. Apply requirements knowledge and techniques to create functional and non-functional requirements for a software system.
3. Apply high and low level design ideas to create an object-oriented design of a software system.
4. Use good design and programming ideas to implement individual and team software systems in compiled OOP languages.
5. Apply white and black box testing techniques and tools to individual and team software development.
6. Use UML class diagrams (and sequence diagrams) to capture aspects of system design and/or requirements (domain).
7. Use practical software development tools, including version control systems, automated build tools, and testing tools.

CSCI 4580. Compilers and Automata Transition
3 Credits (3)
Methods, principles, and tools for programming language processor design; basics of formal language theory (finite automata, regular expressions, context-free grammars); development of compiler components. For Computer Science graduate students only; cannot be used in a students program of study. Taught with CSCI 3730.
Prerequisite: At least a C in (CSCI 2210 or CSCI 4545), in (CSCI 2220 or CSCI 4550), in (CSCI 2230 or CSCI 4555), or consent of instructor.
Learning Outcomes
1. Understand the language theory concepts of regular languages, context free languages, regular expressions, context free grammars, and formal language hierarchy.
2. Use Thompson's construction to convert from regular expression to NFA, and subset construction to convert from NFA to DFA.
3. Apply recursive descent parsing in programming a parser of a small grammar.
4. Understand the ideas in LL and LR parsing of context-free language classes.
5. Understand and use table-driven top-down (LL(1)) and bottom up (SLR) parsing to parse a sentence.

CSCI 4980. Senior Project
4 Credits (4)
Capstone course in which Computer Science majors work in teams and apply computer science skills to complete a large project. Restricted to: Computer Science majors or Cybersecurity majors.
Prerequisite: At least a C- in CSCI 3730 and CSCI 3710.
Learning Outcomes
1. Apply design and development principles in the construction of software systems of varying complexity.
2. Apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
3. Design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
4. Use current techniques, skills, and tools necessary for computing practice.
5. Analyze a problem, and identify and define the computing requirements appropriate to its solution.
6. Function effectively as teams to accomplish a common goal.
7. Communicate effectively with a range of audiences.

CSCI 4996. Special Topics
1,12 Credits
Topics announced in the Schedule of Classes. May be repeated up to 12 credits.
Learning Outcomes
1. Varies.

CSCI 4999. Senior Thesis
4 Credits (4)
Capstone course in which Computer Science majors apply computer science skills to complete a research project, culminating in a written thesis report. Restricted to: Computer Science majors or Bachelor of Science in Cybersecurity degree.
Prerequisite: At least a C- in CSCI 3730 and CSCI 3710.
Learning Outcomes
1. Apply design and development principles in the construction of software systems of varying complexity.
2. Apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
3. Design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
4. Use current techniques, skills, and tools necessary for computing practice.
5. Analyze a problem, identify, and define the computing requirements appropriate to its solution.
6. Communicate effectively with a range of audiences via presentations and technical reports.

CSCI 5110. Data Structure and Algorithms Transition
3 Credits (3)
Introduction to efficient data structure and algorithm design. Order notation and asymptotic run-time of algorithms. Recurrence relations and solutions. Abstract data type dynamic set and data structures based on trees. Classic algorithm design paradigms: divide-and-conquer, dynamic programming, greedy algorithms. For Computer Science graduate students only. Taught with CSCI 3720.
Prerequisite: At least a C- in (CSCI 2220 or CSCI 4550) and a C- in (CSCI 2310 or CSCI 4560), or consent of instructor.
Learning Outcomes
1. Analyze the growth of functions via asymptotic notation.
2. Evaluate the asymptotic running time of a given algorithm.
3. Solve recurrence relations of the kinds encountered in algorithm analysis.
4. Design algorithms using the divide-and-conquer technique.
5. Design algorithms using the greedy technique.

6. Design algorithms using the dynamic-programming technique.
7. Use and analyze data structures based on trees.
8. Analyze the design, correctness, and time complexity of basic graph algorithms.

CSCI 5140. Database Management Systems I
3 Credits (3)
Database design and implementation; models of database management systems; privacy, security, protection, recovery; taught with CSCI 4140; requires more advanced graduate work than taught with CSCI 4140. Students are expected to have solid knowledge of data structures and discrete mathematics.
Learning Outcomes
1. Utilize the basic concepts of relational database model.
2. Utilize database query languages (e.g. SQL).
3. Identify data integrity and security requirements.
4. Analyze, capture, and model user requirements for building database systems using conceptual models.
5. Design and normalize relational schemas.
6. Apply application development methods to implement a database system.

CSCI 5215. Parallel Programming
3 Credits (3)
Programming of shared memory and distributed memory machines; tools and languages for parallel programming; parallelizing compilers; parallel programming environments; taught with CSCI 4215; requires more advanced graduate work than CSCI 4215. Students are expected to have knowledge of programming and machine organization equivalent to CSCI 2210 and CSCI 2230.
Learning Outcomes
1. Describe existing parallel architectures including shared memory versus distributed memory platforms.
2. Apply basic techniques for organizing parallel computations.
3. Apply basic techniques for performance measurement and theoretical limitations of parallelism.
4. Explain alternative parallel techniques and hardware.
5. Perform performance Analysis of different parallel programming techniques.
6. Program shared memory machines using threads, processes, and the OpenMP library.
7. Program using a message passing paradigm and obtain working knowledge of the Message Passing Interface (MPI).

CSCI 5225. Introduction to Cryptography
3 Credits (3)
The course covers basic cryptographic primitives, such as symmetric, public-key ciphers, digital signature schemes, and hash functions, and their mathematical underpinnings. Course helps students understand basic notions of security in a cryptographic sense: chosen plaintext and chosen ciphertext attacks, games, and reductions. Course also covers computational number theory relevant to cryptography. Consent of Instructor required. Taught with: CSCI 4225. Requires more advanced graduate work than CSCI 4225.
Prerequisite: CSCI 2310 (or equivalent) with a C or better.
Learning Outcomes
1. Describe basic cryptographic primitives, including symmetric ciphers, asymmetric ciphers, digital signatures, message authentication codes, and hash functions.
2. Understand the mathematical, fundamental underpinnings of cryptography, and how to reason about the security of crypto primitives: indistinguishability (IND) properties of ciphertexts, CPA/CCA games, and reductions to fundamental math assumptions.
3. Be able to discuss number theory/algebra underpinning the design of cryptographic primitives, in some depth.

CSCI 5235. Cellular Networks and Mobile Computing
3 Credits (3)
This course will offer a solid introduction to major global wireless standards and comparisons of the different wireless technologies and their applications and examine each technology and how to utilize several different systems for the best results. Taught together with CSCI 4235. A basic understanding of Computer Networks is preferable as a course prerequisite.
Learning Outcomes
1. Understand user associations and routing in a cellular/mobile network.
2. Develop insight into interaction of elements within the cellular/mobile core.
3. Understand the concept of end-to-end delivery of a packet and/or signal.
4. Develop an understanding of what happens with the hand-off at each step along the communications path.
5. Be able to explain differences in core architecture between different generations of cellular and mobile network technologies.

CSCI 5240. Software Reverse Engineering
3 Credits (3)
This class provides students with fundamental experience in software reverse engineering with a focus on malware reverse engineering. Students will learn operational security for safely analyzing untrusted code in a sandbox environment. Students will learn control flow integrity attacks, binary control flow analysis, and how to analyze live program behavior. Taught together with CSCI 4240.
Learning Outcomes
1. Students will learn how malware behaves, spreads, and is controlled.
2. Students will learn how to safely analyze malware in controlled environments.
3. Students will learn how malware seeks to hide in systems.
4. Students will learn to perform static analysis of binaries using simple tools.
5. Students will learn how malware obfuscates itself to avoid analysis, including using crypto packers, polymorphism, and sandbox detection.
6. Students will learn to perform decompilation and control-flow analysis of binaries using Ghidra.
7. Students will learn to dynamically analyze malware in a sandbox environment while observing network traffic, resource consumption, and system calls.
8. Students will learn to detect malware running with operating-system level permissions (rootkits).
9. Students will learn memory forensics techniques to detect malware hidden within benign processes.

CSCI 5245. Computer Networks I
3 Credits (3)
Fundamental concepts of computer communication networks: layered network architecture, network components, protocol stack and service. Example of application, transport, network and data link layers, protocols

primarily drawn from the Internet (TCP, UDP, and IP) protocol suite; local and wide area networks, wireless and mobile networks, multimedia networks; network management and security; taught with CSCI 4245; requires more advanced graduate work than CSCI 4245. Students are expected to have solid knowledge of data structures, machine-level programming. Knowledge of statistics (at the level of MATH 371 or MATH 470) is recommended.

Learning Outcomes

1. Understand how to break down the Internet into layers of the OSI model and how each layer of abstraction manages complexity.
2. Understand how data is encoded at the physical layer over copper, fiber, and RF, and the importance of framing and collision avoidance.
3. Understand the concept of packet switching networks, switch fabrics, the ARP, the DHCP, OSPF, and NAT.
4. Understand Internet organization and governance including IANA, ASes, IXPs, ISPs, CAs, and the BGP.
5. Understand the TCP/IP paradigm, including flavors of self-clocking, congestion control, the need for ports, and the end-to-end argument.
6. Understand common application-layer protocols including HTTP(S), FTP, SMTP, etc.
7. Understand the security and privacy guarantees and non-guarantees of TLS, and how they are achieved.
8. Understand the inherent consensus challenges of networked computing, and classical solutions such as the NTP and Lamport Clocks.
9. Write networking program in C that implements an application-layer protocol, directly using system calls and managing memory.

CSCI 5250. Human-Centered Computing
3 Credits (3)

Covers iterative, human-centered interface design, including prototyping and evaluation. Basics of graphic design and visualization. Taught with SCI 4250. Requires more advanced graduate work than CSCI 4250 with an emphasis on studying recent research in human-computer interaction. Students are expected to have knowledge of software engineering equivalent to CSCI 3710.

Learning Outcomes

1. Describe, analyze, and/or critique a device interface using a design vocabulary.
2. Enact a human-centered process of interaction design: gather data; develop a data-driven design; iterate design through testing; and evaluate results.
3. Conduct human-computer interaction research by proposing, developing, and conducting experiments; analyzing data; and developing synthesized results.
4. Communicate design and evaluation with presentations, demos, and reports.
5. Implement a variety of interaction techniques.

CSCI 5255. Digital Game Design
3 Credits (3)

An introduction to digital game design. Topics include design, development, and playtesting of games. The course is structured to use team-based learning. Taught with CSCI 4255. Requires more advanced graduate work than CSCI 4255 with deeper attention to a team game project.

Learning Outcomes

1. Describe, analyze, and/or critique games with a consistent vocabulary.
2. Design, develop, and playtest games.
3. Understand the formal systems of games.
4. Communicate game designs through demonstrations and presentations.

CSCI 5260. Visual Programming
3 Credits (3)

Design and implementation of programs using visual (i.e. dataflow or diagrammatic) programming techniques, with an emphasis on real-time data processing. Students will learn how to design visual programs, including how to handle cycles and state maintenance, and will learn to process audio, video, and other data using visual programs. Students must be in graduate standing to enroll. Taught with CSCI 4260. Requires more advanced graduate work than CSCI 4260.

Learning Outcomes

1. Develop software in graph-based visual environments.
2. Understand flows of control in visual programming environments.
3. Use signals, digital and analog, to drive software.
4. Communicate software design and evaluation with presentations, demos, and reports.

CSCI 5305. Bioinformatics
3 Credits (3)

Introduction to bioinformatics and computational biology. Computational approaches to sequences analysis, protein structure prediction and analysis, and selected topics from current advances in bioinformatics; taught with CSCI 4305; requires more advanced graduate work than CSCI 4305. Students are expected to have a knowledge of algorithms and data structures equivalent to CSCI 3720 or exposure to Biology (equivalent to BIOL 2310 or BIOL 311).

Learning Outcomes

1. Explain the biology motivation of a bioinformatics question.
2. Formulate a computational problem and its solution to address a molecular biology question.
3. Implement basic bioinformatics algorithms such as sequence alignment, pattern matching, and genome assembly.
4. Evaluate the performance of a bioinformatics algorithm on real data sets.
5. Argue the correctness of a bioinformatics algorithm.
6. Analyze the complexity of a bioinformatics algorithm.

CSCI 5310. Bioinformatics Programming
3 Credits (3)

Computer programming to analyze high-throughput molecular biology data including genomic sequences, bulk and single-cell transcriptome, epigenome, and other omics data. Quality control, library size normalization, confounding effect removal, clustering, statistical modeling, trajectory inference, and visualization. Taught with CSCI 4310. Requires more advanced graduate work than CSCI 4310.

Learning Outcomes

1. Write R scripts and functions to manipulate biological sequences, genome annotation, and gene expression data.
2. Perform high-throughput data analysis with established R packages.
3. Detect differential gene expression on RNA sequencing data.
4. Perform single-cell RNA sequencing data analysis (quality control, library size normalization, confounding effect removal, modeling).
5. Assess statistical significance of analytical results.
6. Create automatic data analysis pipeline to link multiple software packages.

CSCI 5405. Artificial Intelligence I
3 Credits (3)
Fundamental principles and techniques in artificial intelligence systems. Knowledge representation formalisms; heuristic problem solving techniques; automated logical deduction; robot planning methods; algorithmic techniques for natural language understanding, vision and learning; taught with CSCI 4405; requires more advanced graduate work than CSCI 4405. Students are expected to have strong knowledge of algorithms and data structures (at the level of CSCI 3720).
Learning Outcomes
1. Use various search algorithms commonly used in problem-solving.
2. Use methods for solving constraint satisfaction problems.
3. Use propositional and first-order logic to represent knowledge.
4. Use logical inference methods to derive conclusions from a knowledge base.
5. Use adversarial search for game-playing agents.
6. Analyze the different search strategies.
7. Design and Implement heuristic search for problem-solving.

CSCI 5410. Computer Graphics I
3 Credits (3)
Languages, programming, devices, and data structures for representation and interactive display of complex objects. Taught with C S 476. Requires more advanced graduate work than CSCI 4410. Students are expected to have knowledge of compilers design and software engineering equivalent to CSCI 3730 and CSCI 3710.
Learning Outcomes
1. Techniques used in three-dimensional graphics.
2. Computer Graphics lightning and shading.
3. Client-server graphics using WebGL.
4. Geometric and Solid modeling.
5. Computer Graphics implementation algorithms.

CSCI 5415. DATA MINING
3 Credits (3)
Techniques for exploring large data sets and discovering patterns in them. Data mining concepts, metrics to measure its effectiveness. Methods in classification, clustering, frequent pattern analysis. Selected topics from current advances in data mining. Students are expected to have a preparation in Discrete Mathematics and Data Structures equivalent to C S 272 and CSCI 2310. Requires more advanced graduate work than CSCI 4415. Taught with: CSCI 4415.
Learning Outcomes
1. Explain and recognize different data mining tasks such as data pre-processing, visualization, classification, regression, clustering, association rules, and anomaly detection.
2. Apply classical data mining / machine learning algorithms for classification, clustering, association rules, and anomaly detection.
3. Evaluate and compare the performance of different data mining / machine learning algorithms.
4. Utilize data mining algorithms to analyze data in real applications using a data mining tool.

CSCI 5420. Applied Machine Learning I
3 Credits (3)
An introductory course on practical machine learning. An overview of concepts for both unsupervised and supervised learning. Topics include classification, regression, clustering, and dimension reduction. Classical methods and algorithms such as linear regression, neural networks, support vector machines, and ensemble approaches. Recent techniques such as deep learning. Focused on applying of machine learning techniques in application domains. Taught with: CSCI 4420. Requires more advanced graduate work than CSCI 4420.
Learning Outcomes
1. Implement and utilize different data processing techniques.
2. Differentiate and assess several dimension reduction techniques.
3. Utilize several classifiers (SVM, Decision tree, k-Nearest Neighbor, and logistic regression) and differentiate their advantages and disadvantages.
4. Explain and demonstrate regression analysis.
5. Describe and illustrate clustering approaches.
6. Apply ensemble learning approaches.
7. Implement several neural network classifiers, including deep learning models.

CSCI 5425. Introduction to Deep Learning
3 Credits (3)
The course covers basic concepts of neural networks which include transition of classical machine learning to hierarchical feature learning, feedforward networks, regularization, optimization, hyperparameter tuning, deep convolutional networks and their applications in computer vision, deep sequence models, and deep generative models. Taught with C S 383. Requires more advanced graduate work than C S 383.
Prerequisite: At least a C- in C S 272 or CSCI 1220, and CSCI 2310, or consent of instructor.
Learning Outcomes
1. Have significant familiarity with different state-of-the-art theories and practices of deep learning.
2. Be able to apply deep learning to a variety of tasks suitable for data science-based projects of academia and industry.
3. Understand much of the current literature on the topic, review papers, and extend their knowledge through further study.
4. Design and evaluate novel deep learning models.
5. Train and test deep learning models on real-life and benchmark datasets using Python libraries such as TensorFlow and PyTorch.

CSCI 5430. Graph Data Mining
3 Credits (3)
The course covers graph terminology, representation, and techniques to extract patterns from large graph data. The topics include random and scale-free graph generation, link analysis (PageRank), graph representation learning, graph neural networks, deep graph generation, community detection, frequent subgraph mining, graph classification, influence maximization, and knowledge graph mining. Taught with CSCI 4430. Requires more advanced graduate work than CSCI 4430.
Prerequisite: At least a C- in C S 272 or CSCI 1220, and CSCI 2310, or consent of instructor.
Learning Outcomes
1. Have significant familiarity with different state-of-the-art theories and practices of graph data mining.
2. Graph representation and graph querying using graph manipulating toolbox/library.
3. Use random and scale-free graph models to generate graphs and visualize complex network properties.
4. Apply algorithms such as PageRank, spectral clustering, and non-negative matrix factorization.
5. Implement graph representation learning algorithms and graph neural network.

6. Understand much of the current literature on the topic, review papers, extend their knowledge through further study, and present findings of the papers.

CSCI 5435. Text Mining and Natural Language Processing
3 Credits (3)
This course is an introduction to text mining and natural language processing (NLP). It covers NLP techniques for extracting insights from unstructured text data. Topics include text classification, semantic textual similarity, topic modeling, sentiment analysis, text summarization, text generation, and machine translation.
Prerequisite: At least a C- in CSCI 2220/C S 272 or C S 463/CSCI 4550 and C S 278/CSCI2310 (or C S 465/CSCI 4560).
Learning Outcomes
1. Describe and apply techniques for text processing, text representation, and text modeling.
2. Describe and apply machine learning algorithms for text mining and NLP tasks such as text classification, semantic textual similarity, topic modeling, sentiment analysis, text summarization, text generation, and machine translation.
3. Utilize Python and popular libraries for implementing NLP-based applications.
4. Evaluate the performance of text mining and NLP algorithms.

CSCI 5440. Generative Artificial Intelligence
3 Credits (3)
Covers the theory and applications of generative artificial intelligence. Concentration will be on specific topics such as large language models, adversarial neural networks, neural symbolic computing, and inductive logic programming. Taught together with CSCI 4440.
Prerequisite: At least a C- in CSCI 4405 or CSCI 5405.
Learning Outcomes
1. Understand the theoretical foundation of generative AI tools.
2. Understand the strengths and weaknesses of generative AI tools and identify appropriate tools for a given application.
3. Utilize advanced generative AI tools such as multi-modal LLMs for problem-solving and developing practical applications.
4. Understand the ethical consequence of using generative AI tools.

CSCI 5505. Analysis of Algorithms
3 Credits (3)
Techniques for design and analysis of algorithms; time and space complexity; proving correctness of programs. Particular algorithms such as sorting, searching, dynamic programming. NP complete problems. Students are expected to have knowledge of algorithms and data structures equivalent to CSCI 3720.
Learning Outcomes
1. Prove algorithm correctness by loop-invariant.
2. Prove an algorithm to be incorrect by counterexamples.
3. Develop efficient divide-and-conquer algorithms.
4. Design and analyze binary search tree algorithms.
5. Construct dynamic programming solutions.
6. Prove the correctness of dynamic programming solutions by contraposition.
7. Traverse graphs efficiently.
8. Find paths in graphs efficiently.
9. Determine if a problem is NP-Complete or NP-Hard. 1
10. Basic concepts of quantum computing.

CSCI 5510. Automata, Languages, Computability
3 Credits (3)
Regular and context-free languages, pushdown and finite-state automata, Turing machines, models of computation, halting problems. Students are expected to have knowledge of algorithms equivalent to CSCI 3720.
Learning Outcomes
1. Describe the language accepted by an automaton or generated by a regular expression or a context-free grammar.
2. Design automata, regular expressions and context-free grammars accepting or generating a certain language.
3. Prove properties of languages, grammars, and automata with formal mathematical methods.
4. Convert between equivalent deterministic and non-deterministic finite automata, and regular expressions.
5. Convert between equivalent context-free grammars and pushdown automata.
6. Define Turing machines performing simple tasks.

CSCI 5605. Operating Systems II
3 Credits (3)
Advanced topics related to operating system principles, guided by the current literature. Students are expected to have knowledge of computer architectures and operating systems equivalent to CSCI 4230 and CSCI 4120.
Learning Outcomes
1. Further an understanding of the principles of operating systems.
2. Develop insight into process management and scheduling issues.
3. Understand memory management operation.
4. Develop an understanding of file system implementation and of multiple levels of hardware support and management.
5. Develop a deep understanding of the concepts of cooperating processes, including communication, synchronization, and deadlock (detection and avoidance).
6. Be able to evaluate operating system features.
7. Develop an understanding of the distributed operating system environment.

CSCI 5750. Artificial Intelligence II
3 Credits (3)
Covers advanced theory and application of artificial intelligence. Concentration on several specific research areas, such as knowledge representation, problem solving, common-sense reasoning, natural language understanding, automated tutoring systems, learning systems. Students are expected to have knowledge of artificial intelligence equivalent to CSCI 4405.
Learning Outcomes
1. Apply selected planning algorithms in solving problems.
2. Identify problems where knowledge representation and reasoning techniques are applicable.
3. Be able to apply answer set programming in problem solving.
4. Be aware of various advanced research topics in Artificial Intelligence.

CSCI 5810. Advanced Software Engineering
3 Credits (3)
Advanced tools and methods for developing large software systems. Topics include object-oriented modeling and design, component architectures, templates and generic programming, software configuration and revision control, static and dynamic analysis tools, model, checking, advanced testing, and verification. Students are

expected to have knowledge of software engineering equivalent to CSCI 3710.

Learning Outcomes

1. Be able to explain modern software development process ideas.
2. Be able to apply agile software development techniques in a project.
3. Be able to specify, design, and develop a complex software system in a team.
4. Be able to properly utilize both black box and white box testing techniques.
5. Be able to explain how unsound and incomplete formal methods can aid in system verification and validation.
6. Be able to utilize sound and complete formal methods to prove properties of a system.

CSCI 5820. Database Management Systems II
3 Credits (3)

Advanced data models and abstractions, dependencies, implementations, languages, database machines, and other advanced topics. Students are expected to have knowledge of data base management systems equivalent to CSCI 4140.

Learning Outcomes

1. Analyze storage and file structures of an RDBMS.
2. Analyze and apply indexing techniques of an RDBMS.
3. Analyze query evaluation approaches of an RDBMS.
4. Analyze the mechanisms of transaction management in an RDBMS.

CSCI 5840. Computer Networks II
3 Credits (3)

Advanced topics in computer networks. Covers advanced topics in networking, with emphasis on wireless, and IP networks. Students are expected to have knowledge of computer networks equivalent to CSCI 4245 and statistics equivalent.

Learning Outcomes

1. Understand design of link layer protocols.
2. Understand challenges and implementations for multimedia streaming.
3. Be able to use basic security constructs in the networking context.
4. Understand the concepts of edge and cloud computing.
5. Understand the concepts and challenges of Internet of Things.
6. Learn concepts of distributed networking.
7. Learn and evaluate future internet architectures.

CSCI 5860. Algorithms in Systems Biology
3 Credits (3)

The course will introduce important algorithms and computational models used in systems biology to study molecular mechanisms for cellular dynamics, processes, and systems. Cellular processes, such as metabolism and signal transduction, are studied as systems and networks quantitatively from high throughput molecular measurements. The topics include molecular biological systems, network alignment, model simulation, network inference, model optimization, and hybrid models. Students will be able to construct models and analyze their properties in the context of molecular biological systems. Students are expected to have knowledge of algorithms and data structures equivalent to CSCI 3720.

Learning Outcomes

1. Create mathematical representation of biological systems.
2. Infer biological network topology from observed omics data set.
3. Simulate the behavior of a biological system using a mathematical model.
4. Characterize behaviors of biological systems.
5. Estimate parameters in a biological system model.
6. Validate a model's statistical relevance given observed data.

CSCI 5991. Special Research Problems
1-6 Credits (1-6)

Faculty-supervised investigation, to culminate in a written report. Maximally 6 credits can be applied to the student program of study. Written agreement with faculty supervisor is the required consent. May be repeated up to 18 credits.

Learning Outcomes

1. Research experience for graduate student.

CSCI 5994. Master's Project
1-6 Credits

Project-oriented capstone course to be completed by Master of Science students under supervision of their advisor. Maximum of 6 credits may be applied toward Master of Science degree. Restricted to Computer Science majors. May be repeated up to 6 credits.

CSCI 5996. Special Topics
1-6 Credits

Topic announced in the Schedule of Classes. May be repeated up to 6 credits.

CSCI 5999. Master's Thesis
1-6 Credits (1-6)

Thesis to be developed by Master of Science Students under supervision of their advisor. May be repeated up to 6 credits.

Learning Outcomes

1. Varies.

CSCI 6991. Pre-dissertation Research
1-15 Credits

Pre-dissertation research. May be repeated up to 88 credits.

CSCI 7000. Doctoral Dissertation
1-15 Credits

Dissertation. May be repeated up to 88 credits.