Learning Objectives for Computer Science Courses Spring 2010

Contents

1 COSI 12b Advanced Programming Techniques

The course will introduce students to object oriented programming using Java. It assumes that students know the basics of scalar types (integers, stings, booleans) and fundamental control structures in procedural programming (loops, assignment statements, conditional expressions). It will focus on more sophisticated features such as design of classes, interfaces, packages and APIs. It will also cover the basic principles of software design, testing, and collaborative programming. It will finally include a short introduction to the Java Collection Framework and the Java API.

Learning objectives

The objective of this class is to:

- Cover issues related to the definition, creation and usage of classes, objects and methods.
- Discuss the principles of inheritance and polymorphism and demonstrate though problem analysis assignments how they relate to the design of methods, abstract classes and interfaces.
- Provide the foundation of good programming skills by discussing keys issues to the design of object-oriented software, including programming design patterns, automatic documentation techniques and programming testing.
- Cover the basics of creating APIs as well as allow students to explore the Java Abstract Programming Interface (API) and Java Collection Framework through programming assignments.
- Discuss basic principles and tools of collaborating programming (versioning systems, code review) and study their usage through group programming projects.

Expected Learning Outcomes

Upon completion of this class, students should be able to:

- Understand the concept of OOP as well as the purpose and usage principles of inheritance, polymorphism, encapsulation and method overloading.
- Identify classes, objects, members of a class and the relationships among them needed for a specific problem.
- Create Java application programs using sound OOP practices (e.g., interfaces and APIs) and proper program structuring (e.g., by using access control identifies, automatic documentation through comments, error exception handling)
- Use testing and debugging tools to automatically discover errors of Java programs as well as use versioning tools for collaborative programming/editing.
- Develop programs using the Java Collection API as well as the Java standard class library.

2 COSI 21A Data Structures and Algorithms

This course focuses on the design and analysis of algorithms and the use of data structures. Through the introduction of the most widely used data structures employed in solving commonly encountered problems (e.g. lists, trees, and graphs), students will learn different ways to organize data for easy access and efficient manipulation. Algorithms to solve classic problems (e.g. searching, sorting, hashing, graph algorithms, etc) will be presented, as well as classic algorithm design strategies (e.g. divide-and-conquer and greedy algorithms). Computational complexity theory will be introduced for studying the efficiency of the algorithms covered in the course.

Learning Objectives

Students who complete this course will be able to:

- Write programs that use data structures such as: arrays, linked lists, stacks, queues, trees, hash tables, and graphs.
- Compare and contrast the cost and benefits of dynamic and static structure implementations.
- Choose the appropriate data structure for modeling a given problem.
- Describe the concept of recursion and give examples of its use, identifying the base case and the general case of a recursively defined problem.
- Compare iterative and recursive solutions for elementary problems.
- Determine when a recursive solution is appropriate for a problem.
- Determine the time and space complexity of simple algorithms and recursively defined algorithms.
- Implement both a greedy and a divide-and-conquer algorithm to solve problems.
- Implement the most common sorting algorithms.
- Design and implement an appropriate hash function.
- Design and implement a collision-resolution algorithm for a hash table.
- Solve problems using the fundamental graph algorithms, including depth-first and breadth-first search, topological sort, minimum spanning tree algorithm, and single-source shortest path.

3 COSI 30A: Introduction to Theory of Computation

Lectures will cover an general introduction to the theory of computation, including regular sets and finite automata, context-free languages, Turing machines and undecidability, and computation complexity (NP-complete problems, etc.). The course centers on the basic theory and philosophy about the nature of computation, what is possible, and what resources are required. Students should have already taken a course in discrete mathematics. Assignments with have problems sets requiring reasoning about the concepts presented in class (little or no programming is required).

Learning objectives

To gain a fundamental understanding of the power and limits of basic models of computation, and to gain comfort with associated proof techniques.

Expected learning outcomes

- The notion of a regular set and its representation by DFA's, NFA's, and regular expressions.
- The notion of a context-free language and its representation by context-free grammars and push-down automata.
- The notion of a universal model of computation and its representation by a Turing machine.
- The notion of an undecidable problem.
- Basic understanding of computational complexity and the P=NP problem.

Expected skill development

- Basic proof techniques, such as proof by induction and diagonalization.
- Practical applications of regular expressions and context-free grammars.
- Basic approaches for accessing the complexity of computational tasks, and comfort with the basic notion of reducing one computational problem to another.
- Experience with giving a presentation of technical material.

4 COSI 31a: Computer Structures and Organization

Course Objectives and Outcomes

This course is an introduction to computer systems organization and operating systems. The objectives of the course are for you to learn three things:

- The first thing is how operating systems and, more generally, computers work. The goal is to demystify the interactions between the software you have written in other courses and hardware. A student graduating with a CS degree should be able to describe the chain of computer system events that occurs from when a user hits the reboot button to when user's program runs, reading input and displaying results, from the register level to the operating system level to the application level. This is important philosophically, but it is also of practical interest to developers who need to figure out how to make a system do what they want it to do.
- The second thing is for you to learn the core ideas in operating systems: resource sharing, virtual addressing, memory protection, scheduling, concurrent programming, file systems, transactions, etc. Such ideas are often best explained as abstractions that some software layer (usually the operating system) provides above imperfect hardware to make that hardware usable by programmers and users. The intent is for you to understand such abstractions well enough to be able to synthesize new abstractions when faced with new problems.
- Many of the ideas and abstractions that we will cover are relevant not only to OS kernels but also to many large-scale systems. Thus, a third goal of this course is to enhance your ability to understand, design, and implement such systems.

Method

- Lectures and reading assignments from the textbook will cover the concepts and techniques.
- Homework problem sets will test and improve your knowledge of the material. The intent is for you to use these as mini exams to assess your knowledge of the course material and evaluate your performance.
- Perhaps the most valuable part of this class will be the programming assignments. They will require you to apply in practice the concepts covered in the lectures and will require you to hold ground defending your design decisions. You will gain experience with implementing in Java advanced synchronization and file systems algorithms. The projects will be done individually and in two-person teams, providing collaborative project experience.
- There will be one midterm and a final exam testing the mastery of the material covered in the course. Unless stated otherwise, both the midterm and final exam will be closed book and will cover material from lectures, readings, problem sets, and the projects.
- As with most technical courses, besides ability and motivation, it takes time to learn and master the subject. Expect to spend an additional 10 to 15 hours a week outside of class time on the average.

5 COSI 113b: Artificial Life

Artificial Intelligence (AI) has set for itself the goal of understanding and simulation human cognition using computer programs, under a slogan that the mind is nothing more than a physical symbol system. Artificial Life (ALIFE) is a younger field which has set for itself the goal of understanding and simulating living systems, under a slogan that life precedes intelligence.

Alife combines several strands complex systems, self-organized complexity, evolution and robotics, with simulated and actual machinery. There is no good textbook yet for the field, so we will read original literature. After a couple of initial lectures, the class will read and discuss papers from the field, with members taking turns leading the class and providing a printed summary/review, handed out to everyone before class.

A major part of the grade derives from a creative term project of your own choosing, which should result in a program, experimental data, and a 10 page double-spaced paper with references.

The Areas of ALIFE we will learn about are:

- Genetic Algorithms and Genetic Programming
- Algorithmic Chemistry
- Assembly Language Based Systems
- Cellular Automata and self-reproduction
- Game models, Prisoners Dilemma
- Co-evolutionary Learning
- L-Systems and other Generative Models
- Artificial Creatures and ALIFE Robots

Learning goals

- Develop knowledge of major elements of the field of artificial life;
- Carry out independent research project;
- Improve both written and oral communication skills.

6 COSI 123A: Statistical Machine Learning

Statistical machine learning tools are essential to analyzing large-scale data sets for mining patterns/rules, inferring causal relationships, making predictions, and so on. This course is designed to focus on learning from data (both numerical and categorical) using statistical analysis techniques and deal with the issues of designing algorithms, techniques, and systems which automatically improve with experience. This course will give students a thorough grounding in the methodologies, technologies, mathematics and algorithms currently needed by research in learning with data. Hands-on training will be provided. Students will be required to engage in experiential learning to practice what they learn in the classroom. Skills learned in this course will be useful for basic and applied research in the fields of bioinformatics, computational neuroscience, data mining, economics, pattern recognition, statistical natural language processing, and so on. This course is listed in the Quantitative Reasoning Program.

Learning objectives

Students are expected to master the following quantitative reasoning skills:

1. Collect and effectively utilize quantitative information.
2. Use mathematical models to express causal relationships and infer them from data.
3. Express clearly facts, ideas, opinions, and beliefs in a variety of formats (such as tables, graphs, charts, and text).
4. Design and carry out in silico experiments.
5. Evaluate different machine learning techniques (e.g., robustness, sensitivity, specificity, advantages, limitations, etc.) by comparing and assessing their computational results.
6. Determine which learning techniques are appropriate to a particular problem domain.

Examples of how Statistical Machine Learning can be applied to other fields in Computer Science

1. Natural Language Processing: word sense disambiguation, probabilistic parsing, statistical machine translation, text retrieval.
2. Computer Graphics: collect data to train complex mathematical models for rendering human motion, natural phenomena, artistic style, and so on.
3. Image Processing and Pattern Recognition: content-based image retrieval.
4. System: analyze/extract the patterns of system utilization (historical trending), prioritize resources intelligently, make systems respond dynamically based on both real-time business service information and historical trending.

7 COSI 125a: Human Computer Interaction

Upon completion of this course, the student will have designed a human computer in- teraction. Students will also acquire a base of knowledge that is sufficient to read technical articles in conferences like CHI, and journals like Human-Computer Interaction (HCI) and Transactions on Computer Human Interaction (TOCHI). The term project enables students both to practice the collection of methods that compose the cycle of interaction design and to develop collaborative skills. A presentation of a technical paper to the rest of class under "conference-like" conditions improves each student's communications skills for presenting technical material. Students will learn how to do a plugin to the Thunderbird email application during two in-class labs that are done in pairs.

The theme for the course this semester will be email.

The last part of the course focuses on design and the human part of interaction between humans and computers. The middle part of the course presents methods for the development of effective, efficient, and learnable technology mediated activities. The last part of the course is run like a workshop. Students will share their term projects, helping each other develop a technology-mediated activity. During the last part of the course students will also be doing short presentations on technical papers.

Learning objectives

- Discuss why user-centered product development is important. Explain why both individual human models and social models are important to consider in design of human computer interaction.
- Use vocabulary for analyzing human interaction with software: perceived affordance, conceptual model, mental model, metaphor, interaction design, feedback, and so forth.
- Explain principles for design of user interfaces, such as learnability, flexibility, and robustness.
- Summarize common interaction styles.
- Define a user-centered design process that explicitly recognizes that the user is not like the developer or her acquaintances.
- Gather requirements for a user interface, using both task analysis and interview/survey with a user.
- Identify from requirements analysis functional requirements and usability requirements and create a specification for a user interface based on requirements.
- Create prototype using specification.
- Describe various evaluation methods, including: observing, surveying, or interviewing a user, cognitive walkthroughs, expert-based analysis, Keystroke Level Model (KLM) analysis, and comparing a given user interface to a set of guidelines or standards to identify inadequacies.
- Create a simple application that supports a graphical user interface, for either the Web or a windowing system.
- Participate in a team project for which some interaction is face-to-face and other interaction occurs via a mediating software environment.

8 COSI 127b: Introduction to Database Systems

Course Objectives:

This course is an introduction to data management with Database Management Systems (DBMS). Students in this course will learn the requisite language, design and analysis skills to perform the functions of a database administraror (DBA) which include: formulation of database queries in various (theoretical and practical) query languages, logical and physical database design, application of techniques to ensure recovery and high availability, and profiling configured systems according to a variety of performance measures. As well, students will learn about the data structures (e.g., B+-trees, file structures) and algorithms (e.g., query optimization) found in most systems, and apply this understanding to the tuning of configured systems to best meet specified performance objectives. This course is intended to complement COSI 128a which focuses on design and implementation techniques used to build Database Management Systems.

Expected Outcomes

Students who complete this course will be able to perform the following tasks:

1. design and implement a database for any specified domain according to well-known design principles that balance data retrieval performance with data consistency guarantees,
2. formulate data retrieval queries in SQL and the Relational Algebra and Calculus,
3. describe the semantics of a SQL query in set-theoretic terms,
4. design experiments that perform a comprehensive analysis of the performance of a database system,
5. apply an understanding of the algorithms used by various database components to predict the time required by the database to perform designated tasks, and
6. apply an understanding of the algorithms used by various database components to tune the database system by modifying the underlying logical or physical design.

9 COSI 133b: Internet & Society

Prerequisite: sophomore standing. This course may not be repeated for credit by students who have taken COSI 33b in previous years. An interdisciplinary survey of the Internet. Taught by a team of professors from several different departments, the course content will vary from year to year. Some particular topics to be covered are the architecture of the Internet (and the implications this has on its regulation), intellectual property, privacy, censorship, e-commerce, online education, and research. Usually offered every year.

Course Learning Objectives

The goal of this course is to explore the internet and its impact on society from several different perspectives. In particular, we will study the hardware, software, languages, and protocols that support the web, The languages used to create web applications, the security issues with web applications and the measures that can be taken to enhance security, the legal issues arising from use of the web with a focus on copyright and patent law, the phenomenon of peer production (e.g wikipedia), and internet business models. We conclude the course with a five week section of student presentations and discussions of emerging web technologies and we apply the reasoning skills we have developed to these particular new technologies.

Expected Learning Outcomes

After successful completion of this course, students should be able to:

- give a clear explanation at the level of hardware devices communicating messages using IP routing of how a browser displays a webpage after the user types in a URL and hits submit;
- understand the role of computer languages and protocols in the workings of the web and in particular to be able to explain the roles of HTTP, HTML, CSS, Javascript, SQL and other languages in web applications;
- describe the most common internet security weaknesses and explain how they allow the attacker to gain access to and/or corrupt private data or to otherwise reduce the effectiveness of a web application.
- describe the most common defenses against online attacks and explain how they work
- understand the benefits and problems with modern copyright and patent law as it applies to the internet
- discuss in detail particular internet technologies which allow large numbers of users to collaborate in building an information resource, and to describe the benefits and drawbacks of this approach to building information resources.
- give examples of the most common business models supporting internet technology today and explain the main expenses and sources of revenue for those technologies.
- analyze in depth a current internet technology in terms of its underlying technology, security model, user interface, the market it serves, its business model, the extent to which it is customizable by users, and the possible disruptive effects it could have on the market.

Expected Skill Development

During this course students will gain experience in

- giving an effective 10-20 oral presentation to a group of around 50 people
- producing a 5-10 minute video combining screen captures, web-browsing voice-overs, video blogging, and possible captioning
- analyzing the quality of a web resource
- writing a research paper as a web page where all claims have links to well-argued supporting evidence

- reviewing the work of others in a thoughtful and useful manner
- writing effective blog entries and responses

10 COSI 146a: Fundamentals of Computer Systems

Even if you have no intention to build systems for living after graduation, you may still need to plan a web site, roll out a financial aplication, or advise management on IT strategy. So you need to understand systems. CS146a will teach you the design principles underlying computer software and hardware systems: techniques for controlling complexity; networks and distributed systems; atomicity and coordination of parallel activities; recovery and reliability; privacy of information; impact of computer systems on society. The emphasis will be on long-lasting ideas rather than low-level mechanics. The lectures will define the concepts. The case studies of working systems and outside reading in the current literature will provide comparisons and contrasts. The hands-on assignments will poke at real systems from outside, the lab will provide experience with building systems from inside. Upon completion of this course, the student will be able to understand a computer system and evaluate its key aspects with respect to performance, reliability and security. In addition, the students will gain proficiency in reading and analysing profesional literature, and gain experience in applying in practice some of the design principles learned in class.

MISSION: Teach students approaches, concepts, abstractions, and techniques that underly the design and implementation of computer software systems.

METHOD: Lectures cover the main concepts and techniques. The lecture material is based on the Jerry Saltzer and Frans Kaashoek book ”Principles of Computer Systems Design”. - Recitations go a level deeper by analysing and evaluating the design of important software systems in class, and by comparing and contrasting case studies of successful and unsuccessful systems. The recitations are based on a (changing) list of selected papers in the literature. - Hands-on assignments provide students the opportunity to poke at real systems from outside. - Lab assignments teach students to come to grips with the complexities arising in real systems, from inside. Currently, the lab is focused on enginering scalable web servers. - Student presentations: in certain years, we offer an independent topic study and in-class group presentations to teach students topic research and oral communication skills. This year we will experiment with a novel presentation tool, prezi. Topics are matched to student interests. The class is loosely modelled after MIT 6.033 Computer System Organization Class, adapted to Brandeis Liberal Arts Colledge Computer Science curriculum and extended with a lab component.

11 COSI 217b: Natural Language Systems

Natural language problems often require systemic solutions that involve complex interactions between different natural language components. For example, an Information Extraction system typically includes the extraction of person names, organizations, locations, events and time expressions from natural language text, the detection of all text units that refer to specific semantic classes and the linking of these text units that refer to the same object, as well as the determination of semantic relations between pairs of entities. These natural language components in turn require underlying natural language technologies such as part-of-speech tagging, phase chunking and syntactic parsing. The objective of this course is to provide an in-depth exploration of one particular type of natural language system. Upon completion of the course, the student will have designed and implemented a working system, drawing from linguistic knowledge, computational techniques and statistical modeling learned during the course. The student will also be required to make a technical presentation and write a technical article on his/her system, acquiring technical communication skills in the process.

12 COSI 236: Principles & Practice behind the formation of a Technology (Software) based Start-Up

Learning objectives

The goal of this course is to provide attendees with a working knowledge of the various facets, practices and principals of the Extreme Programming approach. Upon completion of the course you will understand how Xtreme programming and Entrepreneurship go hand in hand, and how Xtreme programming creates the opportunity for the formation of new successful companies. Finally, upon completing the course, students will they have the knowledge to assess how XP can fit into their current development environment, and some basic experience through the course exercises to begin participating in XP activities.

What You Will Learn? When you complete the course you will have achieved the following:

Entrepreneurship Concepts

- You will understand the sources of competitive advantage in forming a technology based startup
- You will know how to evaluate an idea in terms of its potential success as product in the market place.
- You will understand the basics principles of a successful business plan, and you will learn how to write a Business Plan Executive Summary
- You will understand how to build a successful executive team.

Software Engineering concepts

- You will understand the software development life cycle, and the fundamentals of software engineering including Waterfall Model, the evolutionary model, and the iterative-incremental development model.
- You will know how to read and write use cases and user stories
- You will know how to manage requirements
- You will know how to perform basic problem analysis and design activities
- You will be able to write tests for your code
- You will know how to plan a project and track progress
- You will understand process elements and be able to reason about their applicability in different situations

Expected Outcomes

This course helps fulfill the following CS outcomes:

- Understand programming language concepts, particularly Java and object-oriented concepts.
- Understand software engineering principles and develop an ability to apply them to software design.
- You will complete a large software project.
- Demonstrate independent learning.
- Demonstrate the ability to locate and use technical information from multiple sources.
- Demonstrate an understanding of professional ethics.
- Participate in a class or project team.
- Demonstrate the ability to communicate effectively in speech.
- Demonstrate the ability to communicate effectively in writing.