

COURSE DESCRIPTION

Department and Course Number: CS 128

Course Coordinator: Amar Raheja, Assistant Professor of Computer Science

Course Title: Introduction to C++

Total Credits: 4

Current Catalog Description

Basic concepts of computer software and programming. Data types, expressions, control structures, functions, file and stream I/O. Use of pointers and dynamic storage allocation. Structured and abstract data types. Problem-solving techniques. 4 lectures.

Prerequisites: MAT 105 and 106 with grade of C or better, or consent of instructor.

Textbook

Deitel and Deitel. *C++: How to Program*, Prentice Hall, 2000.

References

Stroustrup, B. *The C++ Programming Language*, Second Edition, Addison-Wesley 1995.

Savitch, W. *Problem Solving with C++: The Object of Programming*, Third edition, Addison-Wesley, 2001.

Goals

An introduction to the basic concepts of programming and its use for problem solving. Learning good programming principles and practices using the C++ language. Good knowledge of the syntax and semantics of C++ and its compilation and execution.

Prerequisites by Topic

Number System
Functions
Polynomials
Systems of equations
Trigonometric functions

Topics

Computer Science overview (1 hour)
Hardware overview (1 hours)
Software overview (2 hours)
Ethical and social concerns (1 hours)
Problem solving and algorithms (4 hours)
Simple programming examples in C++ (2 hours)
Computer logon, file system, use of editor (2 hours)
Primitive types in C++ (2 hours)
Variables, literals, and expressions (2 hours)
Control statements (4 hours)
Functions (Includes Recursive functions) (5 hours)
Introduction to arrays (3 hours)
Strings (2 hours)
Text and File input, output (4 hours)
Introduction to classes and objects (2 hours)
Tests (3 hours)

Laboratory Projects

Several in-class lab exercises to gain familiarity with Windows 2000. Use the Visual C++ editor to write and build files containing the sample programs shown in the handout. Compile and run the programs. Modify the programs. (2 weeks)

Write a program that sums a sequence of integers. Assume that the first integer read specifies the number of values remaining to be entered. Your program should read only one value per input statement. A typical input sequence might look like (1 week)

5 100 200 300 400 500

The factorial of a positive integer n is equal to the product of integers from 1 to n . Write a program to evaluate the factorials of integers from 1 to 5. Print the result in a tabular format. (1 week)

Write a program that uses a function to find if a given integer is a perfect number. Use this function in the program to display all the perfect numbers between 1 and 1000. (1 week)

Write a program that uses the Sieve of Eratosthenes method to determine if a number is prime. Use a single dimensional array for this problem and display the prime numbers between 1 and 1000. (1 week)

Write a program to count the vowels and letters in the text file (the file is neural.txt). Read text a character at a time until you encounter end-of-file. Print out the number of occurrences of each of the vowels a, e, i, o and u in the text, the total number of letters, and each of the vowels as an integer percentage of the letter total. Display the results in a tabular format. (2 weeks)

COURSE DESCRIPTION

Department and Course Number: CS 130

Course Coordinator: Gilbert Young, Associate Professor of Computer Science

Course Title: Discrete Structures

Total Credits: 4

Current Catalog Description

Fundamental topics for Computer Science, such as logic, proof techniques, sets, basic counting rules, relations, functions and recursion, graphs and trees. Prerequisite: MAT 105 with a grade of C or better, or consent of instructor.

Textbook

Gersting, J.L. *Mathematical Structures for Computer Science*. 4th Edition, Freeman, 1999.

Goals

Introduction to mathematical concepts and problem solving techniques that provide foundations for Computer Science.

Prerequisites by Topic

Good understanding of College Algebra

Topics

Mathematical Logic
Proof Techniques
Sets and Combinatorics
Functions and Relations
Graphs and Graph Algorithms

Laboratory Projects

None

CSAB Category Content

	core	advanced
Theoretical Foundations	3	
Algorithms	1	
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture		

Theoretical Content

90% of time devoted to theoretical material

COURSE DESCRIPTION

Department and Course Number: CS 140

Course Coordinator: Barry I Soroka, Professor of Computer Science

Course Title: Introduction to Computer Science

Total Credits: 4

Current Catalog Description

Basic concepts of Computer Science, including overview of hardware and software. Ethical and social impacts of computing. Problem-solving methods. Programming in a high-level language. Written essay required. Prerequisite: MAT 114 with a grade of C or better, or concurrent enrollment in MAT 114, or consent of instructor.

Textbook

David M Arnow & Gerald Weiss, *Introduction to Programming Using Java: An Object-Oriented Approach. Java 2 Update*. Reading MA: Addison-Wesley, 2000. isbn 0-201-61272-0

Computer Science Department. *CS 140 Handbook*. 1998.

Goals

An introduction to the field of computer science and some appreciation for the scope of its subject matter. A solid foundation in problem solving and in good programming principles and practices. Familiarity with the Java language and its compilation and execution.

Prerequisites by Topic

Number systems
Functions
Polynomials
Systems of equations
Mathematical induction
Trigonometric functions

Topics

Hardware overview (2 hours)
Software overview (2 hours)
Computer Science overview (1 hour)
Ethical and social concerns (2 hours)
Problem solving and algorithms (3 hours)
Simple programming examples in Java (2 hours)
Computer logon, file system, use of editor (3 hours)
Primitive types in Java (2 hours)
Variables, literals, and expressions (3 hours)
Control statements in Java (5 hours)
Strings (2 hours);
Introduction to arrays (4 hours)
Introduction to classes and objects (4 hours)
Text input, output (3 hours)
Tests (2 hours)

Laboratory Projects

1. Write a Java application which reads a filename (e.g. `test`) and which creates two new files, `test.line1` and `test.line2`, containing the first and second lines of the file `test`, respectively.
2. Write a class `Person` with instance variables `lastName`, `firstName` and `college`. Provide two constructors. Provide methods `setCollege` and `getCollege`. Provide a method `cross(Person)` which returns a new `Person` with the `firstName` of the message recipient and the `lastName` of the parameter. Write a `toString()` method for `Person`.
3. Write a static `read` method for `Person` which will read the appropriate information from the terminal and return a newly-created `Person` object.
4. Write a program which reads a `String`, converts it to an `int`, and then prints its square.
5. Write a class `Complex` which models complex numbers.
6. Add instance variables to class `Person` which indicate the `Person`'s low and high temperature thresholds. Write a method `judgeTemp(int)` which takes a temperature and turns "Too hot" or "Too cold" or "OK".
7. Use conditional statements to implement a `toString` function for a class with instance variables `quantity` and `species`. Your `toString` should be able to differentiate the following cases:
 - singular vs plural — 0 dogs, a dog, 3 dogs;
 - regular vs irregular plural — 2 dogs but 2 mice;
 - nouns beginning with vowels — a dog vs an elephant.

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms	1	
Data Structures		
Software Design	1	
Programming Concepts	2	
Computer Architecture		

Written Communications

Students write essays about their prior experiences with computers and their reasons for wanting to major in Computer Science.

COURSE DESCRIPTION

Department and Course Number: CS 141

Course Coordinator: Gilbert Young, Associate Professor of Computer Science

Course Title: Introduction to Programming and Problem-Solving

Total Credits: 4

Current Catalog Description

Design, implementation, documentation and testing of programs in an object oriented language. Modularization and reusability of software. File I/O, graphic user interfaces, and exception handling. 4 lecture/problem-solving. Prerequisite: CS 140 and Math 114 with grades of C or better, or consent of instructor.

Textbook

Arnow and Weiss, Introduction to Programming Using Java: An Object-Oriented Approach - Java 2 Update, Addison-Wesley, 2000.

Goals

A solid foundation in problem solving by computer, and in good programming principles and practices. Fuller comprehension of the role of classes, the use of methods, the declaration and manipulation of data. Appreciation of the role of arrays, vectors, and strings; the uses of recursion; the handling of exceptions. Experience with the process of defining objects and the creation of reusable classes. Broader understanding of input and output using the keyboard and display and using files.

Prerequisite by Topic

Introduction to computers and problem solving
Basics of Java
Java control structures
Introduction to use of arrays
Introduction to classes and objects

Topics

Control structures (2 hours)
Objects and classes (4 hours)
Encapsulation and visibility (2 hours)
Methods, parameter passing, overloading (6 hours)
Arrays and vectors (3 hours)
Strings (2 hours)
Recursion (3 hours)
Inheritance and polymorphism (4 hours)
Exceptions (3 hours)
Threads (1 hour)
File I/O (2 hours)
Abstract classes and methods (2 hours)
Applets and graphics (4 hours)
Tests (2 hours)

Laboratory Projects

Write a program that reads in student information from a file. The information for each student is a name, status, and amount purchased for the books. Calculate the average purchasing amount for each student and print it out. If particular students have purchased more than average, write a message saying "above average". Otherwise, write a

message saying “below average”. Print all the student information for the output of your program. (1.5 weeks)

Write a program for Set operations. Add two methods, complement (Set s1, Set s2) and symmetricComplement (Set s1, Set s2), to the Set class given in the book, pages 292 – 300. (1.5 weeks)

Write a program that implements add method, subtract method, multiply method, quotient method, remainder method, and log-base-n method using only increments and decrements. (2 weeks)

Write a program that gets two command-line arguments, say *filename1* and *filename2*. Each string stands for a file name. The program uses the Selection-Sort algorithm to sort the file named *filename1* in ascending order and print out the sorted list. The program performs the same action to the file named *filename2*. Finally the program merges two sorted files into one single sorted file. (1.5 weeks)

Write a program for comparing the performance of the recursive binary search method, *bsearch*, and the recursive linear search method, *lsearch*. (1.5 week)

Write a program that implements a class named Employee, and two subclasses of Employee, named Faculty and Staff. The program overrides the print method in each class to display the class name, the employee’s name and detailed information for the individual objects. (2 week)

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms	1	
Data Structures	1	
Software Design	1	
Programming Concepts	1	
Computer Architecture		

COURSE DESCRIPTION

Department and Course Number: CS 210

Course Coordinator: Salam N. Salloum, Associate Professor of Computer Science

Course Title: Computer Logic

Total Credits: 4

Current Catalog Description

Boolean algebra with applications to computers and logic design. The Arithmetic Logical Unit, logical properties of flip-flops and sequential machines. Applied projects. 4 lectures/problem-solving. Prerequisite: CS 130 with a grade of C or better, or consent of instructor.

Textbook

Charles H. Roth Jr., *Fundamentals of Logic Design*, Fourth Edition. PWS Publishing Company, 1995.

Charles H. Roth Jr., *Logic Aid* (software and user manual). PWS Publishing Company, 1995.

Reference

Mano, M. Morris and Charles R. Kime. *Logic and Computer Design Fundamentals*, Second Edition, Prentice-Hall, 1997.

Goals

Upon the completion of this course, the student shall acquire the following knowledge and skills

Number systems and symbol representations in computer

Binary arithmetic

Boolean algebra and its application in software and hardware

Techniques of analyzing, designing and simplifying logic circuits (combinational and sequential)

Most important SSI, MSI, LSI and VLSI components

Using the CAD software tool LogicAid

Hands-on experience of designing some basic logic circuits using integrated circuit chips.

Prerequisite by Topic

Number systems

Functions

Polynomials

Systems of equations

Mathematical induction

Trigonometric functions

Topics

Number system and binary arithmetic (2 hours)

Boolean algebra and combinational circuits (4 hours)

Applications of Boolean algebra (4 hours)

Karnaugh map (4 hours)

Important combinational circuits: multiplexers, decoders, ROM, PLA, adders, and comparator (4 hours)

Modular Design of combinational circuits (4 hours)

Sequential circuit and flip – flops (4 hours)

Counters and shift registers (2 hours)

Analysis of sequential circuits (2 hours)
 State graph, state table, transition table, and sequential circuit design (4 hours)
 Some important sequential circuits:
 sequence detector, vending machine, and digital clock (2 hours)
 Addition and subtraction of unsigned and signed integers (2 hours)
 Tests (2 hours)

Laboratory Projects

Designing combinational circuits using the Heathkit Digital Design Experimenter and SSI integrated circuit chips (2 weeks).

Using the tutorial of CAD software tool LogicAid to find the essential prime implicants, prime implicants, a minimal sum of product, and a minimal product of sums of Boolean expression (2 weeks).

Designing combinational circuits using Heathkit Digital Design Experimenter and an IC chip of the multiplexer (1 week).

Using the CAD software tool LogicAid to design counters and sequence detectors (1 weeks)

Using the CAD software tool LogicAid to design a digital watch (2 weeks).

CSAB Category Content

	core	advanced
Theoretical Foundations	2	
Algorithms		
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture	2	

COURSE DESCRIPTION

Department and Course Number: CS 240

Course Coordinator: Salam Salloum, Associate Professor of Computer Science

Course Title: Data Structures and Algorithms I

Total Credits: 4

Current Catalog Description

Abstract data types. Searching and sorting. Linked lists. Stacks. Queues. Sets. Analysis of algorithms. Sequential files. Prerequisite: CS 130 and CS 141 with grades of C or better, or consent of instructor.

Textbook

Thomas A. Standish, *Data Structures in Java*. Addison-Wesley, 1998

Reference

Mark Allen Weiss, *Data Structures and Algorithm Analysis in Java*. Addison-Wesley, 1999.

Goals

A broader understanding of the approaches to the organization of data and the algorithms for manipulation of that data. An appreciation for the relationships between data structures and algorithms and some of the performance issues involved. Generic programming. Advanced recursion. Experience with sequential file I/O.

Prerequisite by Topic

Programming capability
Classes and objects
Methods, parameter passing
Encapsulation
Arrays, vectors, matrices

Topics

Interfaces, abstract classes, and generic programming (4 hours)
ADT list, logical representation and physical representation,
and sequential and linked organization of list (4 hours)
One-way linked list (6 hours)
Advanced recursion (4 hours)
Analysis of algorithm (4 hours)
Priority queue (2 hours)
Sequential file (2 hours)
Searching and sorting (2 hours)
Stack, queue, and applications (6 hours)
Circular linked list, two-way linked list, general list (4 hours)
Test (2 hours)

Laboratory Projects

Write a program that implements the ADT SortedList of integers using an array (2 weeks).

Redo Project#1 for the generic data Comparable and test it for several data types (2 weeks).

Redo Project#2 using the linked representation of list (2 weeks).

Write a recursive program for a check-writing system that prints the word equivalent of a check amount (2 weeks).

Write a program that visualizes the recursive computation of the Fibonacci sequence (2 weeks)..

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms	1	
Data Structures	1	
Software Design	1	
Programming Concepts	1	
Computer Architecture		

COURSE DESCRIPTION

Department and Course Number: CS 241

Course Coordinator: John Fisher, Professor of Computer Science

Course Title: Data Structures and Algorithms II

Total Credits: 4

Current Catalog Description

Trees, graphs, hash tables. Random access and indexed files. Prerequisite: CS 240 with a grade of C or better, or consent of instructor.

Textbook

Thomas A Standish, *Data Structures in Java*. Addison Wesley, 1998.

Goals

Familiarity with a wide range of tree and graph structures and their uses. Building data structure libraries and creating documentation for same. Understanding of various representation strategies and experience in the implementation of some search and traversal algorithms. Introduction to the use of hash tables and to string storage and manipulation. Understanding of the applicability of various data structures to file organization and management.

Prerequisite by Topic

Basic data structures and algorithms
Ability to analyze problems and devise solutions
Programming maturity
Inheritance and polymorphism
Stacks, queues, linked lists
Basic sorting, searching
Tree concepts and terminology
Text file input-output

Topics

Language features (2 hours)
Software library organization (3 hours)
Abstract data types (2 hours)
Trees (12 hours)
Graphs (8 hours)
Sorting (8 hours)
Hashing and searching (6 hours)
File concepts: byte io, character io, Java object io, random access (4 hours)
Tests (2 hours)

Laboratory Projects

Define and implement a binary search tree class. Include the class in a reusable library, javalib.util. Design and implement a testing program for the class. (2 weeks)

Define and implement a priority queue class in the javalib.util library. Use a heap tree as storage. Design and implement a FullQueueException in the package. Post the library publicly and test remotely. Students are required to import and test all components of another student's package. (2 weeks)

Implement a Huffman code translator using the priority queue in #2. Program dynamically calculates byte frequencies in file data, builds Huffman code tree, and writes codes and coded data to output file. (Mini-zip utility.) (2 weeks)

Use a graphical diagramming tool to create diagrams, implement a topological ordering algorithm that determines topological sorts for the graphs in the diagrams. (2 weeks)

Write a program to implement and compare several sorting algorithms. (2 weeks)

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms	1	
Data Structures	2	
Software Design	1	
Programming Concepts		
Computer Architecture		

COURSE DESCRIPTION

Department and Course Number: CS 245

Course Coordinator: Chung Lee, Professor of Computer Science

Course Title: Introductory Computer Graphics

Total Credits: 4

Current Catalog Description

Basic concepts in 2-dimensional graphics. Display devices, programming for vector and raster graphics. Language structure and components. 2-dimensional transformations, windowing, clipping, simple hidden line removal. Color graphics principle. (Standards, Animation)

Textbook

Ammeraal Leen, *Computer Graphics for Java Programmers*. Wiley and Sons, 1998.

References

D. Hearn and M.P. Baker, *Computer Graphics: C-Version*. Second Edition. Prentice-Hall (1997).

Goals

Introduction to computer graphics – Hands-on experience with graphics hardware and software and understanding of basic algorithms for generating images.

Prerequisites by Topic

Programming language (CS 140/141 or other language including C)

Topics

Hardware and software principles
Basic algorithms for generating images
Output primitives
Color models
Geometric transformations
Animation
Visualization of Chaos theory
Interactive techniques.

Laboratory projects

Project 1- 2 : 1 week
Project 3 – 4 : 2 weeks each
Project 5 : 3 weeks

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms		
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture		

Theoretical Content

Transformation geometry : 5 hours

Color model : 3 hours
Chaos theory : 3 hours

Problem Analysis

For scene generation, some problem analysis is conducted in class for 1 or 2 hours.

Solution Design

Implement the scene image based on the problem analysis but not done in class but as a part of project.

COURSE DESCRIPTION

Department and Course Number: CS 256

Course Coordinator: Amar Raheja, Assistant Professor of Computer Science

Course Title: C and C++ for Programmers

Total Credits: 4

Current Catalog Description

Data types, expressions, control structures, functions, arrays, file and stream I/O. Use of pointers and dynamic storage allocation. Structured and abstract data types. Class encapsulation, inheritance, polymorphism, templates and exception handling. Problem solving and testing techniques. 4 lectures/ problem-solving. Prerequisite: CS 120 or CS 125 or CS 141 with a grade of C or better, or consent of instructor.

Textbook

Deitel and Deitel. C++: How to Program, Prentice Hall, 2000.

References

Stroustrup, B. *The C++ Programming Language*. Second Edition, Addison-Wesley 1995.

Savitch, W. *Problem Solving with C++: The Object of Programming*. 3rd edition. Addison Wesley, 2001.

Goals

Learn the syntax and semantics of C and C++ programming languages. Develop the working knowledge of programming in C and C++. Appreciate the techniques of problem solving in C and C++. Good knowledge of the syntax and semantics of C++ and its compilation and execution. Begin to use C++ for object-oriented programming.

Prerequisites by Topic

Since the course covers the essentials of C and C++ languages which make extensive use of dynamic data structures involving pointers, students should have the prior knowledge of a programming language and know the basic techniques of problem solving.

Topics

Primitive types in C/C++ (2 hours)
Variables, literals, and expressions (2 hours)
Control statements (3 hours)
Functions and parameter passing (3 hours)
Recursion, Function overloading and default arguments (2 hours)
Pointers (2 hour)
Arrays (static and dynamic) (3 hours)
Strings and input/output (3 hours)
Data Abstraction (structures and classes)(2 hours)
Classes and Objects (2 hours)
Operator overloading (2 hours)
Inheritance (3 hours)
Polymorphism (1 hour)
Inheritance (4 hours)
Templates (3 hours)
Exception Handling (1 hour)
Tests (2 hours)

Laboratory Projects

Several in-class lab exercises to gain familiarity with Windows 2000. Use the Visual C++ editor to write and build files containing the sample programs shown in the handout. Compile and run the programs. Modify the programs. (1 week)

Write a program that is similar to the ones used in an ATM machine. The program will validate a PIN and present the user with a menu similar to that shown in ATMs. Assume a checking and savings account with initial balance of \$1000. (1 week)

Implement a stack using a single dimensional array and a pointer variable. Use the stack to convert numbers between bases. (The numbers systems of interest are binary, octal, decimal and hexadecimal). (2 week)

Develop a class Polynomial that represents a polynomial. Develop this class with proper constructor and destructor functions as well as set and get methods for the inputting and printing the polynomial. Overload the addition, subtraction and multiplication mathematical operators to perform the respective operations using polynomials. (2 weeks)

Create a linked list class template, which allows insertions and deletions only at the front and the back of the linked list. Use this to implement a stack and a queue. (2 weeks)

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms		
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture		

COURSE DESCRIPTION

Department and Course Number: CS 264

Course Coordinator: H C Liu, Professor of Computer Science

Course Title: Assembly Language Programming

Total Credits: 4

Current Catalog Description

Assembly and machine coding of computers. Archetypal von Neumann architecture and cycle of operation, instruction sets, addressing modes, macros and system I/O. Applied programming projects. 4 lecture/problem-solving. Prerequisites: CS 210 and CS 240 with grades of C or better, or consent of instructor.

Textbook

William Jones, *Assembly Language for the IBM PC Family*. Third Edition. Scott/Jones, Inc, 2001.

Reference

Richard Detmer, *80x86 Assembly Language and Architecture*. Jones and Bartlett Publishers, 2001.

Goals

Use assembly language to solve problems.
Interface assembly routines to high level languages.
Efficient control of computer peripherals.
Write and use of interrupt-driven routines.
Define and invoke user-written macros.
Debug assembly programs.
Manipulate bit data.
Create and application of I/O files.
Handle string data effectively.

Prerequisites by Topic

Entering student should have maturity with at least one computer language. Basic computer logic education. Text file I/O. Data structures. Ability to analyze problems and devise solutions.

Topics

Number systems (3 hours)
Basic characteristics of computer operations (3 hours)
Instruction formats and execution (3 hours)
Branch instructions and control structure (3hours)
Computer arithmetic (3 hours)
Character data and bit operations (3 hours)
Input/output (3 hours)
Subprograms (4 hours)
Argument transmission (2hours)
Macro instructions (2 hours)
Floating point representation (2 hours)
Exceptions and interrupts (3 hours)
Tests (2 hours)

Laboratory Projects

Character string I/O.

Macro utility.
Debugger application.
Subprograms and decision making.
Array processing.
Cryptography in string processing.
Class grade report.

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms	1	
Data Structures		
Software Design		
Programming Concepts	1	
Computer Architecture	2	

COURSE DESCRIPTION

Department and Course Number: CS 301

Course Coordinator: Bruce P. HILLAM, Professor of Computer Science

Course Title: Numerical Methods

Total Credits: 4

Current Catalog Description

Error analysis, zeros of a function, systems of linear equations, interpolation, Chebyshev approximation, least squares approximation, numerical integration and differentiation, random processes. 4 lectures/problem-solving. Prerequisites: MAT 208 and MAT 214 and either CS 125 or CS 141 with grades of C or better, or consent of instructor.

Textbook

Cheney & Kincaid. *Numerical Mathematics and Computing*. Fourth Edition. Brookes-Cole, 1999.

Goals

Ability to efficiently solve numerical problems on a computer and to measure the accuracy of the results. Knowledge of methods to estimate and possibly control the errors introduced by fixed-precision number representation. Appreciation of these issues as they relate to finding roots of equations, solving linear systems of equations, curve fitting, numerical integration, numerical differentiation, approximation of functions by splines, overdetermined systems (least squares), minimum of unimodal functions and introduction to simulation using Monte Carlo methods.. Familiarity with the use of some standard numerical analysis program libraries.

Prerequisite by Topic

Calculus
Linear algebra
Basic programming competence
Topics
Integer and floating point representation (2 hours)
Sources of error and error propagation (3 hours)
Solution of linear algebraic equations (7 hours)
Solution of nonlinear equations (5 hours)
Interpolation (4 hours)
Approximation techniques and splines (8 hours)
Numerical integration (4 hours)
Monte Carlo Methods (3 hours)
Numerical differentiation (2 hours)
Tests (2 hours)

Laboratory Projects

Compute change of volume of earth due to 'space dust' by three methods. Compare the results and explain the different answers to demonstrate that correct formulas can yield wrong answers because of error propagation. (1 week)

Given the recurrence relation for the n th modified Bessel function and a table of values, calculate the values and compare with the true values to demonstrate error due to loss of significance and accumulation. (1 week)

Write a program to show that interpolation and approximation are different by demonstrating that interpolation by polynomials over a grid of equidistant points yields a

poor approximating polynomial while an interpolating polynomial over Chebyshev points approximates. Write-up requires that the results are to be reconciled to theory. (1 week)

Rewrite project 3 to conduct a numerical experiment to contrast polynomial interpolation and splines over equidistant points and Chebycheff points to demonstrate that splines approximate using either set. (1 week)

Write a program to compute the interpolating polynomial by solving a linear system of equations. This introduces the Vandermonde matrix and demonstrates how error can propagate when solving an ill-determined linear system. (1 week)

Write a program to show that smaller step size in numerical integration does not get better results. Project also shows better answers are derived using larger step sizes by Romberg Integration. (1 week)

CSAB Category Content

	core	advanced
Theoretical Foundations		2
Algorithms		1
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture		1

COURSE DESCRIPTION

Department and Course Number: CS 310

Course Coordinator: Bruce P. HILLAM, Professor of Computer Science

Course Title: Formal Languages

Total Credits: 4

Current Catalog Description

Regular and context-free grammars and languages, acceptors, ambiguity, closure properties, normal forms, non-deterministic machines, limitations of context-free languages. 4 lectures/problem-solving. Prerequisite: CS 210 and CS 240 with grades of C or better, or consent of instructor.

Textbook

Daniel I. Cohen, *Introduction to Computer Theory*. John Wiley & Sons, Second Edition.

References

H. R. Lewis. and C. H. Papadimitriou. *An Introduction to Formal Languages and Automata*. Prentice-Hall, 1981.

Goals

Students will acquire knowledge and understanding of the various methods for specifying formal languages. They will study properties of regular and context-free languages with emphasis on results that are significant to other areas of Computer Science. Finally students are introduced to the basic Turing Machine and shown that the Turing Machine can recognize both regular and context free languages.

Prerequisites by Topic

Mathematical maturity
Knowledge of fundamental concepts in computer science
Skills in logical reasoning

Topics

Strings and languages, regular expressions (4 hours)
Formal grammars; regular, context-free, and unrestricted grammars; ambiguity (6 hours)
Finite automata, deterministic and nondeterministic; minimizing DFAs (4 hours)
Equivalence of regular expressions, regular grammars, and finite automata (3 hours)
Closure properties of regular languages; proving that languages are and are not regular; the pumping lemma (5 hours)
Pushdown automata, deterministic and nondeterministic (4 hours)
Equivalence of context-free grammars and PDAs (2 hours)
Closure properties of context-free languages; proving that languages are and are not context-free; the pumping lemma (4 hours)
Parsing of context-free languages (3 hours)
Turing Machines (3 hours)
Tests (2 hours)

Laboratory projects

Students were given 2 weeks from the assignment of a project to the due date of the project

Project 1 implements a finite acceptor over the alphabet {a, b} that only accepts non-empty strings where the number of a's is even, and the number of b's has remainder one when divided by three. Students are first required to develop a generic finite acceptor class.

Project 2 implements a push down acceptor for the language Equal, which accepts all strings with an equal number of a's and b's

Project 3 implements a Turing Machine that accepts the language with alphabet $\Sigma=\{a,b\}$ and strings that have three qualities:

Accepted strings are odd in length.

The middle character in an accepted string is a 'b'.

On either side of the middle character is the character 'a'.

CSAB Category Content

	core	advanced
Theoretical Foundations	4	
Algorithms		
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture		

Oral and Written Communications

Each of the 3 assigned projects required a written description of the machine being modeled and how it does it. This description is typically 1-2 pages. Examples are contained in the sample projects.

Theoretical Content

The course is the first introduction to computer theory on an abstract level and lays the practical and theoretical foundation for the theory of languages and compiler theory.

Problem Analysis

Homework is also assigned. All covered machines are analyzed for correctness.

Solution Design

The central theme of this course is the theory of formal languages and how they are derived and recognized.

COURSE DESCRIPTION

Department and Course Number: CS 331

Course Coordinator: Bruce P. Hiram, Professor of Computer Science

Course Title: Design and Analysis of Algorithms

Total Credits: 4

Current Catalog Description

Development of algorithms, top-down structured programming, program correctness, backtrack programming, branch and bound methods, efficient algorithm implementation, algorithm complexity analysis. 4 lectures/problem-solving. Prerequisite: CS 241 and MAT 116 with grades of C or better, or consent of instructor.

Textbook

Bruce P Hiram, *The Design and Analysis of Algorithms: An Intuitive Approach*. Author-published and available on CD-ROM.

References

T H Cormen, C E Leiserson, R L Rivest, *Introduction to Algorithms*. The MIT Press and McGraw-Hill, 1991.

Goals

To investigate the principles and techniques underlying the design of efficient algorithms. We will examine several well-known mathematical problems and design provably good, if not optimal, algorithms for their solution by applying specific algorithm design techniques. Algorithms will be analyzed for correctness and efficiency.

Prerequisites by Topic

Knowledge of asymptotic functions
Skills in data representation
Ability to reason abstractly

Topics

Mathematical preliminaries (2 hours)
Growth of functions (2 hours)
Recurrences (1 hours)
Sorting and order statistics (5 hours)
Divide-and-conquer algorithms (5 hours)
Backtracking and Branch-and-Bound (4)
Greedy algorithms (4 hours)
Dynamic programming (4 hours)
Searching (4 hours)
Graph algorithms (5 hours)
P=NP (2 hours)
Tests (2 hours)

Laboratory Projects

Students were given 2 weeks from the assignment of a project to the due date of the project.

Project 1 considers whether applying any one of 6 heuristics to the ordering of three different data sets in the recursive version of the knapsack problem can reduce the algorithm's computational complexity.

Project 2 implemented the chain matrix multiplication problem recursively and using dynamic programming on chains of 4 sizes. The number of recursions was counted and used to show the basic problem had exponential complexity, and related this to program execution time.

Project 3 solved the knapsack problem using the branch and bound technique to the three data sets from project In this project the number of visited nodes in the state-space tree and the number of recursions from project 1 are contrasted to determine if the branch-and-bound technique is computationally effective and why.

Project 4 analyzes the partition strategy of QuickSort. Two strategies, first object as pivot and random object as pivot as well as a third pivot strategy developed by the student from considerations discussed in class are analyzed by counting object compares. Students must determine which strategy is best and why.

Project 5 requires the student to implement the Ford-Fulkerson network flow algorithm which requires several graphical algorithm techniques

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms		4
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture		

Oral and Written Communications

Each of the 5 assigned projects required a written analysis of the strengths and weakness of the technique or the implications of the data that was generated by the project. The analysis was typically 1-2 pages. Examples are contained in the sample projects.

Theoretical Content

The first three topics lay the theoretical foundation for algorithm analysis in 5 hours. The correctness proofs require knowledge of mathematical induction. These are interspersed with more practical implementation issues throughout the term. Several optimization results used theory to show that optimums had been achieved. The P=NP problem introduces the theory behind the most significant open problem in algorithm theory.

Problem Analysis

All covered techniques are analyzed to determine complexity and optimality.

Solution Design

The center theme of this course is the design and analysis of algorithms. All sections are introduced to the classic techniques of algorithm design.

COURSE DESCRIPTION

Department and Course Number: CS 352

Course Coordinator: Barry I Soroka, Professor of Computer Science

Course Title: Symbolic Programming

Total Credits: 4

Current Catalog Description

Languages for processing symbolic data with emphasis on applications in artificial intelligence. Coverage of Lisp and Prolog. 4 lectures/problem-solving. Prerequisite: CS 241 with a grade of C or better, or consent of instructor.

Textbook

Stuart C Shapiro, *Common Lisp: An Interactive Approach*. New York: W.H. Freeman, 1992.

W F Clocksin and C S Mellish, *Programming in Prolog*. Second Edition. Springer-Verlag, 1984.

Goals

Knowledge of the basic principles of symbolic programming. Ability to write programs of moderate complexity in LISP and Prolog. Familiarity with the concepts and techniques of Smalltalk.

Prerequisites by Topic

Basic and advanced data structures
Text file input-output
Ability to analyze problems and devise solutions
Programming maturity

Topics

The LISP interpreter (3 hours)
Atoms, lists, strings (3 hours)
LISP functions (6 hours)
Recursion (3 hours)
Sequential constructs (3 hours)
Execution and debugging (3 hours)
Input-output (2 hours)
Prolog syntax (3 hours)
Prolog mechanisms (6 hours)
Prolog programs (6 hours)
Tests (2 hours)

Laboratory Projects:

Create a LISP function which finds the first root of a quadratic equation. (1 week)

Write the LISP function which takes in the coefficients of a quadratic equation and returns a list of the two roots. (1 week)

Write a LISP function which takes in the coefficients of a quadratic equation and returns a list of all its real roots. (1 week)

Write a recursive LISP function which takes a list of numbers and returns the sum of their squares. (1 week)

Write a function that does pattern matching with variables. (1 week)

Write the tail recursive function SP which takes a list of numbers and returns a two-element list containing the sum and the product of the input list. Write the tail recursive function SUM which takes two integer arguments and returns the sum of the numbers between them. (1 week)

Write various short functions in Prolog. Write Prolog code to solve problems such as magic squares.

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms		
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture		

COURSE DESCRIPTION

Department and Course Number: CS 356

Course Coordinator: Daisy F. Sang, Professor of Computer Science

Course Title: Object-Oriented Design and Programming

Total Credits: 4

Current Catalog Description

Elements of the object model. Abstraction, encapsulation, modularity and hierarchy. Algorithmic decomposition vs. object-oriented decomposition. Class diagrams, object diagrams, module diagrams, and process diagrams. Comprehensive examples using a case study approach. 4 lectures/problem-solving. Prerequisite: CS 240 or CS 256 with grades of C or better, or consent of instructor.

Textbook

Booch, G., Rumbaugh, J., and Jacobson, I. *The Unified Modeling Language User Guide*, Addison-Wesley, 1999.

References

Page-Jones, M. *Fundamentals of Object-Oriented Design in UML*. Addison Wesley, 2000.

Budd, T. *Understanding Object-Oriented Programming with Java*. Updated Edition. Addison-Wesley, 2000.

Lee, R. and Tepfenhart, W. *UML and C++: A Practical Guide to Object-Oriented Development*. Second Edition. Prentice-Hall, 2001.

Goals

An introduction to the field of object-oriented software design and some appreciation for the scope of its subject matter. A solid foundation in object-oriented programming principles and practices. Familiarity with object-oriented software design methods.

Prerequisites by Topic

Data types
Structures, arrays, and graphs
Language support for objects, classes, polymorphism, and inheritance
Iteration and iterative processes
Top-down and bottom-up algorithm design

Topics

Introduction to OO (2 hours)
Review of classes, inheritance, and polymorphism (2 hours)
Object-oriented design guidelines (2 hours)
Introduction to UML (2 hours)
Classes and objects: identification and refinement (4 hours)
Using CRC cards and responsibility driven approach (4 hours)
Using nouns (2 hours)
Structural modeling (2 hours)
Behavioral modeling (4 hours)
Development process (2 hours)
Case studies of OO system development (2 hours)
Practice of object-oriented programming (4 hours)
Implementing object-oriented design (2 hours)
Team project discussions and presentations (4 hours)

Tests (2 hours)

Laboratory Projects

Team project: design and implementation of an elevator simulation system. Report includes defining a list of actions the system must perform, finding the objects, determining their responsibilities, determining collaborations, walking through scenarios to refine the class design, and team member effort report. Presentation includes the design process, the demo of the implementation with well-defined scenarios to show most of the functionality of the program, and a discussion of the strength and constraints of the work. (4 weeks)

Team project: design and implementation of a personal scheduler for students. Report includes class diagram, use case diagram, normal scenarios, scenarios with exceptions, sequence diagrams for nontrivial use cases, state diagrams for object classes with nontrivial dynamic behavior, activity diagrams for nontrivial operations, and team member effort report. Teams not giving presentation for the first project will do so now and presentation format is defined as above. (4 weeks)

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms		
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture		

COURSE DESCRIPTION

Department and Course Number: CS 365

Course Coordinator: H C Liu, Professor of Computer Science

Course Title: Computer Organization

Total Credits: 4

Current Catalog Description

Fundamental characteristics of logical devices used in computer architecture. Building blocks and organization of digital computers, including the arithmetic/logic, control and memory units, and input/output devices and interfaces. Microprocessor system design. 4 lectures/problems. Prerequisites: CS 264 and PHY 133 with grades of C or better, or consent of instructor.

Textbook

John Hennessy & David Patterson, *Computer Organization and Design*. Morgan Kaufmann.

The TTL Data Book for Design Engineers. Second Edition. Texas Instruments.

H. C. Liu, *Lab Manual for RISC Single-Board Computer*. Cal Poly University Bookstore.

References

The TTL Data Book for Design Engineers. Second Edition. Texas Instruments.

Goals

Students from this course should be able to:

1. Comprehend complete digital processing systems.
2. Master the skills required to understand the internal working of computers at the chip level.
3. Remove the mystery surrounding hardware of computing machines.
4. Build a small computing system based on RISC architecture.
5. Understand designing control systems based on finite state machine or microprogramming principles.
6. Comprehend future trends of computer architectures and their effect upon science and technology development.

Prerequisites by Topic

The course is particularly suited to junior, senior, or graduate students majoring in computer science. Entering students should have a good background in electricity, electronics, computer logic, assembly language programming, and semiconductor circuits.

Topics

Basic logic gates
Logic IC families
Processor architectures
Features of processor components
Data path and control unit
Finite state machine fundamentals
Performance of computers
Microprogramming
Organization of control systems

Laboratory Projects

Emulation of an 8-bit ALU
Emulation of a 4-bit CPU
Construction of a 4-bit computer
Write and run an assembler program for the 4-bit RISC single-board computer

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms		
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture	4	

COURSE DESCRIPTION

Department and Course Number: CS 370

Course Coordinator: Lan Yang, Professor of Computer Science

Course Title: Parallel Processing

Total Credits: 4

Current Catalog Description

The taxonomy of concurrent and parallel systems. Communication and synchronization, multicomputer and multiprocessor systems. Shared-memory and message passing programming paradigms. Parallel problem solving. 4 lectures/problem-solving. Prerequisite: CS 331 with a grade of C or better, or consent of instructor.

Textbook

Wilkinson, B. and Allen, M., Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers. Prentice-Hall, 1999.

Goals

Concepts of concurrent and parallel systems, and of communicating and synchronizing parallel processes. Knowledge of parallel problem solving techniques, including basic parallel algorithms. Implementation of parallel problems on typical parallel systems such as MPI and PVM.

Prerequisite by Topic

Typical sequential algorithms such as sort, search, and matrix related algorithms.
Techniques of sequential algorithm design including partition, divide-and-conquer.
Data structures such as matrices, lists, and trees
Basic concepts of computer systems
Skills in sequential problem solving and programming

Topics

Introduction to parallel processing (2 hours)
Taxonomy of concurrent and parallel systems (4 hours)
Communication and synchronization (8 hours)
Multicomputer and Multiprocessor systems (4 hours)
Shared memory problem solving (4 hours)
Message passing systems (MPI and/or PVM) (4 hours)
Message passing problem solving (8 hours)
Case studies (4 hours)
Tests (2 hours)

Laboratory Projects

Using a message-passing parallel system (PVM or MPI) to perform a series of parallel operations (sum, minimum, maximum etc.) on rows and columns of a matrix. (2 weeks)

Using a message-passing parallel system (PVM or MPI) to implement parallel sort and search algorithms. (2 weeks)

Using a message-passing parallel system (PVM or MPI) to solve a number of numerical problems and to study the performance of parallel programs theoretically as well as practically. (3 weeks)

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms		
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture		

COURSE DESCRIPTION

Department and Course Number: CS 380

Course Coordinator: Robert W. Kerbs, Assistant Professor of Computer Science

Course Title: Introduction to Computer Networks

Total Credits: 4

Current Catalog Description

Network architectures and layered protocols. Network service interfaces; addressing and routing; flow and congestion control. Local and metropolitan area networks. Higher level protocols. 4 lectures/problem-solving. Prerequisite: CS 241 and CS 365 with grades of C or better, or consent of instructor.

Textbook

Larry L. Peterson and Bruce S. Davie, *Computer Networks: A Systems Approach*. Morgan Kaufmann Publishers, 2000.

References

<http://java.sun.com> — Sun Microsystems Java web site.

<http://www.ietf.org/rfc.html> — Internet Engineering Task Force (IETF) Requests For Comments (RFCs) web site.

Goals

Students should gain an overall perspective of how network architectures are designed and built, as well as a detailed understanding of prominent protocol examples at various levels in a layered protocol design. The course should involve hands-on experience with networking in the department's computing laboratories. Graduating students who have taken this course should be considered relatively expert by employers seeking to hire network administrators.

Prerequisites by Topic

Data Structures
Graph Theory
Programming Proficiency
Basic Operating Systems
Basic Computer Design

Topics

Network Architectures—OSI and TCP/IP
Layering and Protocols
Bandwidth / Throughput
Congestion Control
LAN Construction and Ethernet
Routing Algorithms
Domain Name System (DNS)
Network Layer
Internet Protocol (IP)
Physical Layer
Data Link Layer
Transport Layer
TCP, UDP, and RPC
Application Layer
Packet Switching
Cell Switching

Cryptography
 Internetworking
 Authentication Protocols and Kerberos
 Electronic Mail (SMTP, POP, IMAP)
 World Wide Web (HTTP/CGI)
 Wireless Networks
 QoS

Laboratory Projects

Client/Server (TCP Sockets) Round Trip Time (RTT) packet analysis application (2 weeks)
 Client/Server (Datagrams) message encoder/decoder (2 weeks)
 Router table emulation application using TCP Sockets (2 weeks)

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms		
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture		

Oral and Written Communications

The course includes 3 or 4 written homework assignments of typically 4 or 5 pages in addition to exams, quizzes, and commented programs. Written homework is expected to be well-written, without spelling or grammatical errors. The course does not necessarily include oral presentations.

Social and Ethical Issues (Hours of Class Time Spent)

Effect of Modern Networking on Society (1 hour)
 Commercialism and Computer Networking (1 hour)

Theoretical Content

Theories applied in networking include graph theory and queuing theory. Graphs provide a natural model for networking topologies, and graph algorithms (e.g., shortest path and spanning tree construction) appear in many practical designs for packet routing. Queuing theory is relevant in congestion and flow control. Students are expected to have completed CS 241(Data Structures and Algorithms II), which provides coverage of appropriate graph theory and algorithms, and are also expected to have completed CS 365, which includes background queuing theory as it applies to resource utilization in operating systems.

Problem Analysis and Solution Design

Practical networking is the result of some years of evolution involving university and industry efforts. In some ways, it is as much driven by ad hoc standards and practices as by theoretically elegant designs. As such, the student is exposed to many designs that have gained practical favor. For example, students will gain exposure to Internet-style networking, including the design of the Internet Protocol (IP), the design of the Transmission Control Protocol (TCP), and the design of several application-layer protocols (e.g., SMTP, FTP and HTTP).

COURSE DESCRIPTION

Department and Course Number: CS 390

Course Coordinator: Chung Lee, Professor of Computer Science

Course Title: Computer Simulation

Total Credits: 4

Current Catalog Description

Current Catalog Description : Overview of computer simulation. Model building, implementation, validation. Discrete and continuous simulation models. Use of simulations languages such as GPSS, Simscript, SLAM and Dynamo.

Textbook

Paul Fishwick, *Simulation Model Design and Execution*. Prentice-Hall, 1995.

References

Simulation language manuals — GPSS and Dynamo.

Goals

Basic coverage of establishing system models, execution and analysis of the results.

Prerequisites by Topic

Programming competence
Statistical methods and probability

Topics

Modeling concepts
Random number generation
Discrete and continuous modeling methods
Statistical analysis
Applications in computer science and physical sciences

Laboratory Projects

Project 1: 2 weeks
Project 2: 3 weeks
Project 3: 4 weeks

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms		
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture		

Oral and Written Communications

Every student is required to submit at least two written reports of typically 10 pages.

Theoretical Content

Random number generator -- 3 hours
Statistical distribution-- 4 hours
Analysis of data-- 2 hours

Queuing theory-- 3 hours

Problem Analysis

Students form teams (size 4-6) to conduct problem analysis of real world situations and build appropriate models. This is done during in-class discussion and off-class meetings. They spend an average 8 hours per problem.

Solution Design

Results of analysis are translated to model design and eventually to programs.

COURSE DESCRIPTION

Department and Course Number: CS 405

Course Coordinator: H C Liu, Professor of Computer Science

Course Title: Microprocessor Systems

Total Credits: 4

Current Catalog Description

The microprocessor and support integrated circuits (ICs) as a unified system and their programming implications. Study and application of ICs for communications, peripheral adaptors, arithmetic processors, floppy disc and CRT controllers in a system context. 4 lecture/problem-solving. Prerequisite: CS 365 with a grade C or better, or consent of instructor.

Textbook

Barry Brey, The Intel Microprocessors 8086/8088, 80186, 80286, 80386, 80486, Pentium and Pentium Pro Processor. Fifth Edition. Merrill.

H. C. Liu, Lab Manual for 8088-base 16-bit Single-board Computer. Cal Poly Pomona Bookstore.

References

Intel Corporation, Microprocessor and Peripheral Handbook, Vol. 1 & 2.

Intel Corporation, iAPX 86/88, 186/188 User's Manual, Hardware Reference.

Intel Corporation, iAPX, 86/88~ 186/188, User's Manual, Programmer's Reference.

Goals

1. Gain a comprehensive knowledge of the organization, function, and operation of the central processing unit of a microcomputer.
2. Develop the ability to interface a microprocessor with memory units and peripheral devices using modem interfacing IC chips.
3. Integrate programming ability in assembly language and machine language toward the application of system routines in solving practical problems.
4. Improve computing speed by the use of interrupt and direct memory access facilities.
5. Understand and apply I/O and arithmetic coprocessors.

Prerequisites by Topic

Electricity
Electronics
Assembly language programming
Computer logic
Computer organization
Basic semiconductor devices.

Topics

Addressing modes
Instruction set
Coding, assembling, linking, executing assembly language programs
Hardware specification
Memory interface
I/O device interface

Interrupts
Direct memory access
Coprocessors

Laboratory Projects

Machine language programming
Real time clock implementation
Desk calculator implementation
Serial data transmission

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms		
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture		4

COURSE DESCRIPTION

Department and Course Number: CS 408

Course Coordinator: Hsun K. Liu, Professor of Computer Science

Course Title: Programming Languages

Total Credits: 4

Current Catalog Description

Concepts in programming languages. Virtual machines and abstraction. Language processing. Declarations and types. Data abstraction. Control abstraction. Concurrent programming. Programming paradigms. 4 lectures/problem-solving. Prerequisite: CS 264, and CS 310 with grades of C or better, or consent of instructor.

Textbook

Sebesta, *Concepts of Programming Languages*. Addison Wesley, Fifth Edition, 2002.

Reference

Appleby, Doris. *Programming Languages: Paradigm and Practice*. McGraw-Hill, 1991.

Goals

Knowledge of the important concepts of abstraction, information hiding, parameterization, and binding. Appreciation of the major programming paradigms, and of issues such as concurrency. Understanding of the run time representations of programs and data structures and their relation to programming language and computer architecture.

Prerequisite by Topic

Basic and advanced data structures
Programming maturity
Assembly language programming

Topics

Describing syntax and semantics (5 hours)
Data types (4 hours)
Expressions and assignment (4 hours)
Control structures (4 hours)
Subprograms (4 hours)
Concurrency (4 hours)
Data abstraction (5 hours)
Functional and logic programming (4 hours)
List processing and string manipulation (4 hours)
Tests (2 hours)

Laboratory Projects

Using FORTRAN write a program that reads an amount in Arabic numerals and spells it out in English words. (2 weeks)

Implement a class "Complex" class in Java that performs all arithmetic and relational operations for complex numbers. Pay attention to the concepts of "information hiding" and "side-effect". (2 weeks)

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms		
Data Structures		
Software Design		
Programming Concepts		4
Computer Architecture		

COURSE DESCRIPTION

Department and Course Number: CS 420

Course Coordinator: Halina Przymusinska, Professor of Computer Science

Course Title: Artificial Intelligence

Total Credits: 4

Current Catalog Description

Heuristic programming, searching problem spaces, theorem-proving programs, game playing programs, decision-making programs, question answering programs. Consideration of ethical and social dilemmas posed by AI. Technical paper required. 4 lectures/problem-solving. Prerequisite: STA 326 and CS 310 with a grade of C or better, or consent of instructor.

Textbook

Nils Nilsson, *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann, 1998.

Reference

Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

Goals

Understanding of the goals and ramifications of Artificial Intelligence. Familiarity with different approaches to knowledge representation and knowledge manipulation in the search for solutions to complex problems.

Prerequisite by Topic

Good working knowledge of at least one high level programming language.
Ability to develop and implement solutions to (relatively) non-trivial problems.
Ability to work with formal and abstract concepts.

Topics

Artificial Intelligence: history and applications (3 hours)
Search and search strategies (7 hours)
Knowledge representation and inference mechanisms (8 hours)
Commonsense knowledge and reasoning with uncertain information (6 hours)
Logic based planning methods (5 hours)
Neural networks (4 hours)
Learning (3 hours)
Genetic Algorithms (2 hours)
Tests (2 hours)

Laboratory Projects

Write a program which takes as an input a permutation of the numbers 0..8 which represents a configuration of the 8-puzzle and returns the sequence of moves which would produce the solution [1,2,3,8,0,4,7,6,5].

Write a program to solve map coloring problem using minimal conflict and arc consistency algorithms.

Group project: Build an intelligent agent for the Wumpus World.

CSAB Category Content

	core	advanced
Theoretical Foundations		1
Algorithms		1
Data Structures		1
Software Design		1
Programming Concepts		
Computer Architecture		

COURSE DESCRIPTION

Department and Course Number: CS 431

Course Coordinator: H. K. Liu, Professor of Computer Science

Course Title: Operating Systems

Total Credits: 4

Current Catalog Description

Overview of operating systems. Operating system structures. Process management. Concurrency and synchronization. Deadlock. Processor management. Scheduling and dispatch. Memory management. Virtual memory. Device management. File systems. Security, privacy and acceptable use. Technical paper required. 4 lectures/problem-solving. Prerequisite: CS 241 and CS 365 with grades of C or better, or consent of instructor.

Textbook

Abraham Silberschatz, Peter B Galvin, Greg Gagne, *Operating System Concepts*. Sixth Edition. John Wiley & Sons, 2002.

Reference

- F. J. Corbató and V. A. Vyssotsky, *Introduction and Overview of the Multics System*.
- R. C. Daley and P. G. Neumann *A General-Purpose File System For Secondary Storage*.
- Dennis M. Ritchie *The Evolution of the Unix Time-sharing System..*
- D. M. Ritchie and K. Thompson *The UNIX Time-Sharing System*.
- Ronda Hauben *The History of UNIX*.

Goals

Understanding of the role and purpose of the operating system, the history of operating system development, functionality of a typical operating system, the algorithms and data structures employed by operating systems. Experience in the implementation of algorithms. General knowledge of how the basic principles are applied in some specific operating systems.

Prerequisite by Topic

- Basic and advanced data structures
- File organization
- Basic computer architecture
- Ability to analyze problems and devise solutions
- Programming maturity

Topics

- Operating system evolution and objectives (2 hours)
- System architectures (2 hours)
- Operating system structures (2 hours)
- Process management (6 hours)
- Memory management (6 hours)
- Concurrency (4 hours)
- Deadlocks (3 hours)
- Secondary storage management (3 hours)
- File systems (3 hours)
- Protection (3 hours)
- Distributed systems (2 hours)

Ethical and social concerns (2 hours)

Tests (4 hours)

Laboratory Projects

Write a program to simulate CPU scheduling. Implement algorithms for FCFS, SJF, and Round Robin scheduling. (2 weeks)

Write a program to evaluate different solutions to the critical section problem in a simulated multiprogramming environment. (2 weeks)

Write a program to simulate paging activity, allocation algorithms for global, equal, and proportional allocation and page replacement algorithms for FCFS and LRU replacement. (2 weeks)

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms		2
Data Structures		
Software Design		1
Programming Concepts		
Computer Architecture		1

COURSE DESCRIPTION

Department and Course Number: CS 435

Course Coordinator: Chung Lee, Professor of Computer Science

Course Title: Database Systems

Total Credits: 4

Current Catalog Description

Database system fundamentals. System components and architecture. Data models, including Entity-Relationship model, relational model and object oriented model. Theory of database design and data manipulation processes using relational algebra and calculus. Advanced topics including distributed systems, concurrency, and recovery. Technical paper required.

Textbook

Lewis, P, A. Bernstein and M.Kifer (2002). Databases and Transaction Processing: An Application-Oriented Approach. Addison-Wesley

References

C. Lee (2002) CS 435 handout and class notes.

Various online or printed manuals for E-R drawing kits (DDT, SmartDraw, Visio or ERWin), Microsoft Access, Java, C++ and "JDBC API Tutorial and reference."

Goals

Theoretical foundations of database system — both schema design and data manipulation.

Prerequisites by Topic

Data structure – II (CS 241)

Topics

Modern DB system features
Major components of DB system
Case study – registration DB design
ER software tools – ERWIN, DDT, Smartdraw, ...
Object Oriented Data Model (OODM)
Concepts and DDL
DBMS – MS Access and JDBC
Concept, classification
Relational algebra
SQL language
Relational calculus and VQL, MS-Access
Using embedded SQL via JDBC and ODBC
Data Definition and structure from E-R analysis
Dependencies (FD,MFD, JD) and normal forms
OODB — basic concepts and applications
Web-based application — XML
Distributed databases
Decision Support Systems — warehouse and mining
Transaction processing
Query optimization

Laboratory Projects

Project 1 — 2 weeks
Project 2 — 3 weeks
Project 3 — 3 weeks

CSAB Category Content

	core	advanced
Theoretical Foundations		1
Algorithms		
Data Structures		2
Software Design		1
Programming Concepts		
Computer Architecture		

Oral and Written Communications

Every student is required to submit at least 3 written reports of typically 12 pages.

Social and Ethical Issues

We cover the issue of the importance of privacy and security of database toward the end of the quarter. Typically this takes 2-3 hours and we have group discussion on this subject.

Theoretical Content

Relational theory: D, MVD, JD
Normalization, functional dependency cover, key closure
Relational algebra and calculus (domain and tuple)
Object Oriented database model — concepts and implementation
Encryption/decryption standards

Problem Analysis

We examine a realistic real-world situation, and we identify the entities, relationships and operational components. We draw ER diagrams and FD diagrams for this analysis.

Solution Design

We design a relational database based on the analysis we did for the real world scenario.

COURSE DESCRIPTION

Department and Course Number: CS 440

Course Coordinator: Lan Yang, Professor of Computer Science

Course Title: Compiler Design

Total Credits: 4

Current Catalog Description

Lexical analysis, parsing and basic compiling techniques including syntax-directed translation. 4 lectures/problem-solving. Prerequisite: CS 241, CS 264 and CS 310 with grades of C or better, or consent of instructor.

Textbook

Alfred V Aho, Ravi Sethi, Jeffrey D Ullman, *Compilers: Principles, Techniques and Tools*. Addison-Wesley, 1988.

Goals

Knowledge of the theory of DFA-driven lexical analysis, and of top-down and bottom-up syntax analysis including recursive descent, LL, and LR techniques. Implementation of a lexical analyzer and a syntax analyzer. Experience in using lexical and parser generator tools.

Prerequisite by Topic

Ability to write and debug large programs
Data structures such as tables, link lists, and trees
Assembly level pseudo codes
Regular language theory
Context free language theory

Topics

Introduction to compiling (4 hours)
Regular language theory (2 hours)
Lexical analysis (4 hours)
Basic compiling techniques (4 hours)
Context-free language theory (4 hours)
Context-free grammar normalizations (4 hours)
LL parsers (4 hours)
LR parsers (6 hours)
Introduction to syntax-directed translation (6 hours)
Tests (2 hours)

Laboratory Projects

Using a lexical/scanner generator tool to construct a lexical analyzer for a subset of a programming language from a list of regular expressions, each denoting the set of lexemes associated with a terminal symbol. (2 weeks)

Using a parser generator tool to construct a LR parser as well as a syntax-directed semantic analyzer which evaluates attributes of an S-attributed translation scheme during LR parsing. For example, checking of syntax errors as well as limited type errors for a subset of a programming language can be implemented. (3 weeks)

CSAB Category Content

	core	advanced
Theoretical Foundations		1
Algorithms		
Data Structures		
Software Design		2
Programming Concepts		1
Computer Architecture		

COURSE DESCRIPTION

Department and Course Number: CS 441

Course Coordinator: Lan Yang, Professor of Computer Science

Course Title: Advanced Compiler Design

Total Credits: 4

Current Catalog Description

Run-time environments, parsing techniques, intermediate code generation and optimization, object code generation and optimization. 4 lectures/problem-solving.
Prerequisite: CS 440 with a grade of C or better, or consent of instructor.

Textbook

Alfred V Aho, Ravi Sethi, Jeffrey D Ullman, *Compilers: Principles, Techniques and Tools*. Addison-Wesley, 1988.

Goals

In-depth study of bottom-up syntax analysis, including SLR, LR, and LALR techniques. Knowledge of the theory of syntax-directed semantic analysis and its applications to type checking, code generation, interpreter design, and general semantic definition. Implementation of a bottom-up parser, a type checker, and/or an intermediate code generator. Experience in using parser generator tools.

Prerequisite by Topic

Ability to write and debug large programs
Regular language theory
Context free language theory
Lexical analysis techniques
Top-down and bottom-up parsing concepts

Topics

Review of bottom-up parsing concepts (2 hours)
Shift-reduce parsers (4 hours)
LR parsing table construction (8 hours)
Syntax directed translation (6 hours)
Type checking(2 hours)
Run time environments (4 hours)
Intermediate code generation (6 hours)
Code generation (4 hours)
Introduction to code optimization (2 hours)
Tests (2 hours)

Laboratory Projects

Using a parser generator tool to construct a semantic analyzer such as a type checking system from a context-free grammar and syntax-directed definition that defines the syntax and the type system of a subset of a programming language. A lexical analyzer can be provided or built using a lexical analyzer construction tool. (2 weeks)

Using a parser generator tool to construct a syntax-directed semantic analyzer such as a code generator, which evaluates attributes of a syntax-directed definition during bottom-up shift-reduce syntax analysis. (3 weeks)

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms		
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture		

COURSE DESCRIPTION

Department and Course Number: CS 445

Course Coordinator: Robert W. Kerbs, Assistant Professor of Computer Science

Course Title: Advanced Computer Graphics

Total Credits: 4

Current Catalog Description

Advanced concepts in the design of 3-dimensional graphics. Transformations, curve and patch generation, hidden line and surface removal, shading, animation. Interactive graphics applications in CAD/CAM. 4 lectures/problem-solving. Prerequisite: CS 245 with a grade of C or better, or consent of instructor.

Textbook

J.D. Foley, A. van Dam, S.K. Feiner, J.F. Hughes, *Computer Graphics: Principles and Practice*. Second Edition. Addison-Wesley, 1990

Reference

Edward Angel, *Interactive Computer Graphics: A Top Down Approach with OpenGL*. Second Edition. Addison-Wesley, 2000.

Hearn and Baker, *Computer Graphics*. Second Edition. Prentice Hall, 1997.

GLUT Tutorial. (<http://www.lighthouse3d.com/opengl/glut/>).

OpenGL Programming Guide: The Red Book (<http://fly.cc.fer.hr/~unreal/theredbook/>).

Class handouts / optional language manuals for graphics utilities. Java Handbook and C++ manual.

Goals

Continuation of CS 245—advanced concepts of computer graphics and user interface design. Enhancement algorithms for realistic image generation. Exploration of database applications to computer graphics such as CAD/CAM. Further understanding of color models and their application. Free-form curve generation. Further coverage of GUI—design and implementation.

Prerequisites by Topic

All the topics covered in Introduction to Computer Graphics (CS 245)
Calculus and Trigonometry
Understanding one high level programming language
Knowledge of hardware and language access

Topics

Introduction
Review of computer graphics principles
Survey of application areas
Advances in computer graphics systems – Hardware/Software
3-Dimensional coordinate system
Transformations and Animations
Geometry and coordinate systems
Composite transformation
Advanced Graphics libraries and tool kits
3D Computer Graphics pipeline
Curve Generation

Geometry and calculus of curved line generation (Hermite, Bezier, b-spline and Overhouser)
 Comparison of different algorithms
 Hidden Surface Removal Algorithms
 Classification and history
 Object space algorithms
 Image space algorithms
 Clipping
 Shading, Lighting, and Color
 Surface rendering
 Surface smoothing
 Color separation for realism
 Texture Mapping
 Fractal surfaces
 Modeling Hierarchies
 Ray Tracing
 Ray Casting
 Classification of various algorithms
 2-D and 3-D ray tracing
 Advanced Applications
 CAD/CAM
 Database system
 Human Computer Interaction (HCI) principles

Laboratory Projects

The course includes 3 or 4 non-trivial programming assignments.

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms		
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture		

Oral and Written Communications

The course includes 3 or 4 written homework assignments of typically 4 or 5 pages in addition to exams, quizzes, and commented programs. Written homework is expected to be well-written, without spelling or grammatical errors. The course requires one 10-minute oral presentation. Students are evaluated based upon technical merit, completeness, factual accuracy, and writing style (grammar, spelling, formatting).

Social and Ethical Issues

A brief in-class discussion is conducted on the impact of computer graphics on society and the importance of designing interfaces that are accessible. Occasionally, an essay is required to examine the impact of Photo-realistic visualization; it is graded as a part of the Written Communication requirement. Approximately 40-minutes of classroom time is allocated for these activities.

Theoretical Content

Most of the material covered in this class is theoretical in nature. This includes shading and lighting models, 3D computer graphics architectures, modeling, rendering,

calculating surfaces (2D/3D), animation techniques, geometry, calculus, human vision, and psychology. The time allocation among these topics is relatively evenly spaced.

Problem Analysis and Solution Design

Each program is designed, implemented, and analyzed based upon theoretical correctness, practical software and hardware limitations, and efficient algorithmic implementations.

COURSE DESCRIPTION

Department and Course Number: CS 450

Course Coordinator: Halina Przymusinska, Professor of Computer Science

Course Title: Computability

Total Credits: 4

Current Catalog Description

Turing machines, RAM machines, primitive and mu recursion, Godel numbering, Church-Turing thesis, decidability, Markov and Post systems, algorithmically unsolvable problems, intractable problems. 4 lectures/problem-solving.

Textbook

Hopcroft, John E., Motwani, Rajeev, and Ullman, Jeffrey D. *Introduction to Automata Theory, Languages, and Computation*. Second Edition. Addison-Wesley, 2001.

Reference

Lewis, H. R., and Papadimitriou, C. *Elements of the Theory of Computation*. Prentice-Hall, 1981.

Rich, Craig A., *Computability*. Unpublished lecture notes.

Goals

An understanding of abstract formal models of computational problems and algorithms and ability to use them in order to obtain practical results about the inherent difficulty of computational problems and the inherent limitations of algorithms.

Prerequisite by Topic

Set theory
Logic
Formal grammars
Finite automata
Pumping theorem
Pushdown automata
Closure properties

Topics

Models of computation (3 hours)
Turing machines and computers (3 hours)
Computation, enumeration, acceptance, decision problems (2 hours)
Recursively enumerable languages (2 hours)
Recursive languages (2 hours)
Diagonalization (2 hours)
Paradoxes and undecidable problems (4 hours)
Rice's theorem (2 hours)
Church-Turing thesis (2 hours)
Recursive functions (2 hours)
Phrase structure grammars (2 hours)
Formalizing relative computability (6 hours)
Hardness and completeness (4 hours)
Tests (4 hours)

Laboratory Projects

There are generally no computer projects assigned in this course.

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms		
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture		

COURSE DESCRIPTION

Department and Course Number: CS 460

Course Coordinator: Keyu Jiang, Assistant Professor of Computer Science

Course Title: Secure Communication

Total Credits: 4

Current Catalog Description

Public-key systems, digital signatures, ciphers, the Data Encryption Standard, access security, control of information flow. 4 lectures/problem-solving. Prerequisite: senior standing in Computer Science and CS 301 with a grade of C or better, or consent of instructor.

Textbook

Bruce Schneier, *Applied Cryptography*. John Wiley & Sons, 1996.

References

William Stallings, *Cryptography And Network Security: Principles and Practice*. Prentice Hall, 1998.

Goals

Students should learn cryptographic communication protocols for key exchange, authentication, and resource access control, and understand their use in practical secure communication mechanisms. Advance protocols, such as secret sharing, coin flipping, and digital cash, will be discussed and analyzed. Students should be able to identify the weakest security link in a networked system. Several leading cryptographic systems, including DES and RSA will be analyzed and discussed. Students should be able to encrypt and decrypt simple messages using S-DES by hand. Graduating student who have taken this elective should be considered relatively expert by employers seeking computer security specialist.

Prerequisites by Topic

Discrete Structures
Programming Proficiency
Worst-Case Running Time Analysis
Boolean and Propositional Logic
Basic Probability
Basic Computer and Network Architecture
Ability to Analyze Problems and Devise Solutions

Topics

Cryptography Foundations (2 hours)
Substitution and Transposition Ciphers (2 hours)
Secret Key Cryptography (2 hours)
Public Key Cryptography (2 hours)
Key Exchange/Distribution Protocols (3 hours)
Authentication Protocols (3 hours)
DES and S-DES (4 hours)
RSA (2 hours)
Secure Information Management System (4 hours)
Secret Sharing (2 hours)
Fair Coin Flips (2 hours)
Digital Cash (2 hours)
Guest Presentation (1/2 hours)

Hackers, Attacks and Virus (2 hours)

Laboratory Projects

Implementation of SIMS for creation and reading procedure (7 weeks)

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms		
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture		

Oral and Written Communications

Every student is required to submit at least 8 written reports of typically 1-10 pages,

Social and Ethical Issues

Maintaining Privacy in a Globally Networked Society (1 hours)

Who do you trust? (2 hours)

Protest your system from Hackers (1 hours)

Theoretical Content

Perfect Secrecy (1 hour)

One-Way function (2 hours)

RSA public-key cryptography (2 hours)

Zero-knowledge proofs (1/2 hour)

Secret splitting and reconstruction (2 hours)

Problem Analysis

User authentication

Key exchange

Letter Frequency analysis in English context

Secret Sharing

Coin Flips

Digital Cash

Analysis S-DES

Analysis protocols and implementation of SIMS

Solution Design

Implementation for SIMS

An Essay to improve the security or efficiency of SIMS

Encryption and decryption messages by using S-DES

COURSE DESCRIPTION

Department and Course Number: CS 463

Course Coordinator: Mandayam A. Srinivas, Professor of Computer Science

Course Title: Undergraduate Seminar

Total Credits: 2

Current Catalog Description

Technical presentations by students on current developments in computer science. Seminar discussions of ethical, social and economic impacts of technology. Essays on seminar topics. 2 lecture discussions. Prerequisite: senior standing in computer science and a passing score on GWT.

Textbook

None

References

- Kizza, J.M., Ethical and Social Issues in the Information Age. Springer Verlag, 1997.
- Garson, D.G., Computer Technology and Social Issues. Idea Group Publishing, 1995.
- Johnson, D.G. and Nissenbaum, H. (eds.), Computers, Ethics & Social Values. Prentice-Hall, 1995.
- Rosenberg, Richard. The Social Impact of Computers. Academic Press, 1992.
- Forester, T. and Morrison, P. Computer Ethics: Cautionary Tales and Ethical Dilemmas in Computing. MIT Press, 1990.
- Johnson, D. G. Computer Ethics. Third Edition. Prentice Hall, 2001.

Goals

Independent, in-depth research of a specific topic in Computer Science. Heightened student awareness of ethical, social, and economic issues in computer science. Develop students' oral and written communications abilities.

Prerequisite by Topic

- Experience in oral communication as provided by general education courses
Competence in written communication
Maturity in computer science

Topics

- Software piracy
The Electronic Frontier Foundation
Computer Professionals for Social Responsibility
Certification of computer professionals
Computer crime
Computer law
Computer viruses
Computerization of the workplace
Computers and the handicapped
Computers and Privacy
Computer Networks and Security
Patent and copyright issues
Artificial intelligence and robotics: ethical and social issues
Program testing/reliability vs. program cost

Policies for ethical use of computer resources
 Virtual Reality
 Human-computer Interaction
 Optical Computing
 Topics (continued)
 Neural Networks
 Parallel Computing
 Cryptography
 Voice and Speech Recognition
 Genetic Algorithms
 DNA-based and molecular computers
 Robots of the Future
 Computers in Medicine
 Computing in the 21st Century

Students select from the list of topics above and may suggest additional topics. The instructor selects one or more topics for sample presentation. Writing assignments cover additional topics. Each topic covered receives approximately 1 hour of class time.

Laboratory Projects

There are generally no computer projects assigned in this course.

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms		
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture		

COURSE DESCRIPTION

Department and Course Number: CS 480

Course Coordinator: John R. Fisher, Professor of Computer Science

Course Title: Software Engineering

Total Credits: 4

Current Catalog Description

Software engineering process including requirements engineering, specification techniques, design concepts and methods, software testing and integration concepts, verification and validation, quality assurance and configuration management, post-development software evolution and documentation. Prerequisite: CS 241 with a grade of C or better, or consent of instructor.

Textbook

Bruegge, Bernd and Dutoit, Allen H., *Object-Oriented Software Engineering*, Prentice Hall, 2000.

Reference

Fisher, J.R. Jed Development Notes,
<http://www.csupomona.edu/~jrfisher/jeds/jed1.0/JedDevelopmentNotes.html>.

Goals

1. To present a software engineering methodology
2. To complete a significant project using that methodology
3. To introduce visual specification and design tools
4. To learn background of software engineering and to learn key terminology

Prerequisites by Topic

Principles of programming
Data structures
Applied state or transition diagrams (not theory)

Topics

1. Simple project planning and estimation principles including metrics
2. Principles of requirements gathering and refinement
3. Storyboarding techniques for requirements generation
4. Principles of UML.
5. Use Case analysis and diagrams
4. Class hierarchy diagrams, Object diagrams
5. Behavior diagrams
6. Evolutionary/incremental process model
7. Testing principles: numerical methods, object units, integration testing
8. Documentation principles
10. The "Ripple" development method for student projects (instructor's method)

Laboratory Projects

A moderate to large size Java project that implements concepts for some Cal Poly Course. Previous examples include applets demonstrating physics, mathematics, Computer Science and music content. Students often have another instructor as the "client" from whom they must get the requirements.

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms		
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture		

Oral and Written Communication

Every student is required to submit at least 3 written reports of typically 3-5 pages and to make 2 oral presentations of typically 30 minutes duration.

Social and Ethical Issues

Discussion of what projects are appropriate to undertake.

Discussion of privacy issues that are related to the access and data security of a software system.

Frequent discussion about appropriate attribution for content and ideas; fairness of workload of student teams.

Theoretical Content

Algorithm complexity (5%)

State diagrams to analyze system usage (15%)

Analysis and Design

These are the major topics of the course (80%). Please see the Topics section above.

COURSE DESCRIPTION

Department and Course Number: CS 499

Course Coordinator: Daisy F. Sang, Professor of Computer Science

Course Title: Introduction to Distributed Computing

Total Credits: 4

Current Catalog Description

Distributed computation models. Network architecture and internetworking. Distributed objects and remote invocation. Synchronization and distributed algorithms. Processes and threads. Transaction and distributed file servers. Case studies. 4 lectures/problem-solving. Prerequisite: CS 241 with grades of C or better, or consent of instructor.

Textbook

G Coulouris, J Dollimore, T Kindberg, *Distributed Systems: Concepts and Design*. Third Edition. Addison-Wesley, 2001

Reference

H Attiya, and J Welch, *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. McGraw Hill, 1998.

N Lynch, *Distributed Algorithms*. Morgan Kaufmann, 1996.

G Tel, *Introduction to Distributed Algorithms*. Second Edition. Cambridge University Press, 2000.

J Farley, *Java Distributed Computing*. O'Reilly, 1998.

Goals

A recognition of the complexity involved with developing communication software systems. An overall perspective of key distributed programming concepts, techniques, and tools. An introduction to distributed algorithms, synchronization, communications and transactions.

Prerequisites by Topic

Data Structures and Algorithms
Programming Proficiency
Basic Operating Systems
Basic Computer Systems

Topic

Characterizations of distributed systems (2 hours)
Distributed computation models (2 hours)
Network architectures (2 hours)
TCP and UDP (2 hours)
Internet addressing (2 hours)
Java socket programming (4 hours)
Distributed objects and remote invocation (2 hours)
Broadcast, convergecast, leader election (4 hours)
Routing algorithms (4 hours)
Mutual exclusion (2 hours)
Distributed Agreement (2 hours)
Network File System (NFS) (2 hours)
Domain Name System (DNS) (2 hours)
Object replication and consistency control (2 hours)
Recovery and Fault Tolerance (4 hours)

Laboratory Projects

Implementation of a two-way chat application consisting of a server and multiple clients. Java socket-based communication was used. (2 week)

Implementation of a distributed bank with a set of branches. TCP was used for communication between branch server objects and branch GUI objects. (2 week)

Implementation of a distributed bank with message logs for recovery from branch server fail stop failure. (3 weeks)

CSAB Category Content

	core	advanced
Theoretical Foundations		
Algorithms		
Data Structures		
Software Design		
Programming Concepts		
Computer Architecture		

Theoretical Content

Formal message passing network model and distributed algorithms are covered. Fundamental distributed algorithmic problems include broadcast, convergecast, election, resource allocation, mutual exclusion, deadlock detection, consensus and clock synchronization. Complexity measures such as communication complexity and time complexity for distributed algorithms are used for worst-case analysis.

SUPPORT COURSES

COURSE DESCRIPTION

Department and Course Number: MAT 114

Course Coordinator: Bernard Banks, Professor of Mathematics

Course Title: Analytic Geometry and Calculus I

Total Credits: 4

Current Catalog Description

Functions, limits and continuity. Derivatives and applications of Derivatives including max/min applications, L'Hospital's Rule. Introduction to Integration, the Fundamental Theorem of Calculus, the Indefinite Integral integration by substitution. 4 lecture/problems. Prerequisite: Must have satisfied ELM and have achieved the minimum placement score on the appropriate MDT or B or better in MAT 105 and MAT 106 or equivalent or C or better in MAT 112 within two quarters.

Textbook

James Stewart, *Calculus, Early Transcendentals*. Fourth Edition. Brooks/Cole, 1999.

Goals

Students will be familiar with the concept of limit; they will understand the definition of derivative. They will be able to differentiate algebraic and trigonometric functions, exponentials and logs, inverse trig functions, and implicit functions. They will be able to apply differential calculus in solving problems.

Prerequisite by Topic

Number systems
Functions
Polynomials
Systems of equations
Trigonometric functions

Topics

Function, limits and continuity
Derivatives and Differential Rules
Velocity
Interpretations and rates of change and as slope
The derivative as a function
Computations of derivatives
Sum, product, quotient rules
Power rule
Chain rule
Differentiation of trigonometric functions, exponential functions, log functions, inverse functions, implicit functions
Applications of the derivative
Derivative as a graphing tool
Increasing and decreasing, max, min]
Second derivative and concavity
Global vs. local max, min
Rates of change
Related rates

COURSE DESCRIPTION

Department and Course Number: MAT 115

Course Coordinator: Bernard Banks, Professor of Mathematics

Course Title: Analytic Geometry and Calculus II

Total Credits: 4

Current Catalog Description

Applications of the Definite Integral, Calculus of Inverse Functions including trig functions, log and exponential functions, and hyperbolic functions. Integration techniques including substitution, parts, products of trig functions, partial fractions, trig substitution, quadratic forms, improper integrals. 4 lecture/problems. Prerequisite: C or better in MAT 114 or consent of the instructor.

Textbook

James Stewart, *Calculus, Early Transcendentals*. Fourth Edition. Brooks/Cole, 1999.

Goals

Students will understand the definitions of definite and indefinite integrals. They will be able to apply integral calculus in the solutions of problems. They will be familiar with a variety of integration techniques.

Prerequisite by Topic

Limits, continuity
Derivatives
Definite and indefinite integrals
Trigonometric functions

Topics

Definition of definite and indefinite integral
Riemann sums
Properties of integrals
The fundamental theorem
The integral as the net change in a function when the rate of change is known
Elementary anti-derivatives and simple substitutions
Numerical methods
Techniques of integration
Integration by parts
Basic trig integrals
Trig substitutions
Partial fractions
Applications
Area, distance and other net change examples
Volumes by slicing and by shells
Work
Arc length
Moments
Additional Topics
Hyperbolics
Ad hoc substitutions

COURSE DESCRIPTION

Department and Course Number: MAT 116

Course Coordinator: Bernard Banks, Professor of Mathematics

Course Title: Analytic Geometry and Calculus III

Total Credits: 4

Current Catalog Description

Sequences and Series, Polar Coordinates, Parametric equations and Conic Sections. 4 lecture/problems. Prerequisites: C or better in MAT 115 or consent of the instructor.

Textbook

James Stewart, *Calculus, Early Transcendentals*. Fourth Edition. Brooks/Cole, 1999.

Goals

Students will understand the basic concepts of sequences and series. They will be able to evaluate indeterminate limits and improper integrals. They will be able to graph parametric equations and graph functions in polar coordinates.

Prerequisite by Topic

Derivatives
Definite and indefinite integrals
Trigonometric functions

Topics

Improper integrals and L'Hospital's rule
Sequences and series
Infinite sums, convergence
Basic tests for convergence
Geometric series, P-series, alternating series, integral test, ratio test, comparison test, limit comparison test, absolute convergence, conditional convergence
Power series, radius of convergence
Taylor series, Taylor polynomials, Taylor's theorem
Parametric functions and equations
Conic sections, rotation of axes
Polar coordinates

COURSE DESCRIPTION

Department and Course Number: MAT 208

Course Coordinator: Richard Robertson, Professor of Mathematics

Course Title: Introduction to Linear Algebra

Total Credits: 4

Current Catalog Description

Introduction to linear transformations of the plane, vector space of n-tuples, matrix algebra, determinants, systems of linear equations. 4 lecture/ problems. Prerequisite: C or better in MAT 214, or consent of instructor.

Textbook

The choice of Textbook is up to the instructor.

Goals

Students will be thoroughly acquainted with the methods and principles of linear algebra and will gain reasonable skills in their practical applications.

Prerequisite by Topic

Derivatives
Definite and indefinite integrals
Calculus of vector-valued functions
Functions of several variables

Topic

Linear transformations, vector spaces, linear independence (10 hours)
Matrix algebra, inversion of matrices, singular matrices, rank of a matrix (10 hours)
Determinants, permutations, sub-determinants, Laplace's theorem (10 hours)
Systems of linear equations, Cramer's rule, matrix methods for the solution of linear systems (10 hours)

COURSE DESCRIPTION

Department and Course Number: MAT 214

Course Coordinator: Bernard Banks, Professor of Mathematics

Course Title: Calculus of Several Variables I

Total Credits: 3

Current Catalog Description

Introduction to vectors, dot products, cross products, equations of lines and planes. Calculus of Vector Valued Functions including unit tangents, unit normals, and curvature. Introduction to multivariable functions, the Differential Calculus of Multivariable Functions, the chain rule, applications including extreme problems and Lagrange multipliers. 3 lecture/problems. Prerequisite: C or better in MAT 116 or consent of instructor.

Textbook

James Stewart, *Calculus, Early Transcendentals*. Fourth Edition. Brooks/Cole, 1999.

Goals

Students will be familiar with the concepts of differentiation of functions of several variables. They should be able to solve a variety of problems by using the tools of calculus.

Prerequisite by Topic

Derivatives
Definite and indefinite integrals
Infinite series
Functions of several variables

Topics

Definitions of Vector algebra, dot products, cross products

Equations of lines and planes

Calculus of vector valued functions, velocity and acceleration, unit tangents and normals, curvature

Functions of several variables, level curves and surfaces, quadric surfaces, cylindrical and spherical coordinates

Calculus of functions of several variables, limits, partial derivatives, the differential and differentiability, the chain rule, directional derivatives and the gradient, normals and tangent planes, extrema, and Lagrange Multipliers

COURSE DESCRIPTION

Department and Course Number: STA 326

Course Coordinator: Alan Krinik, Professor of Mathematics

Course Title: Statistical Methods for Computer Scientists

Total Credits: 4

Current Catalog Description

Rules of Probability. Discrete and continuous distributions including the multinomial distribution. Sampling distributions. Point and interval estimation. Hypothesis testing. Large and small sample inferences for means, proportions, and variances. Introduction to queuing theory and regression. 4 lecture/problems. Prerequisites: C or better in MAT 214 or consent of instructor. Not open to students required to take STA 330.

Textbook

Miller, I., Freund, J. and Johnson, R. *Probability and Statistics for Engineers*, Fourth Edition Prentice Hall.

Goals

Students are expected to learn the rules of probabilities and become familiar with some common distributions. Students will also be able to do statistical analysis of data, using the procedures covered in the course.

Prerequisite by Topic

Calculus – derivatives and integrals
Sequences and series
Limits

Topics

Basic concepts of probability theory
Sample space, events, rules of probability
Discrete and continuous random variables
Probability distributions and expectations of random variables
Joint distributions
Special distributions
Binomial, geometric, hypergeometric, Poisson, multinomial
Uniform, normal, exponential
Sampling distributions
Central limit theorem
Student's t, Chi-squared, and F-distributions
Estimation and testing hypothesis
Properties of point estimators
Method of maximum likelihood
Confidence intervals for mean, proportion, and variance
Testing hypothesis for mean, proportion, and variance
Queuing theory and regression: (as time permits)
Single server queue (M/M/1)
Simple linear regression
Least squares estimators
Prediction and confidence intervals

COURSE DESCRIPTION

Department and Course Number: PHY 131

Course Coordinator: George Rainey, Professor of Physics

Course Title: General Physics

Total Credits: 3

Current Catalog Description

Fundamental principles of mechanics, vectors, statics, uniform motion, accelerated motion, work and energy, momentum, rotational motion, and fluid mechanics. 3 lecture/problems. Prerequisite: MAT 114. Corequisite: MAT 115 and PHY 131L.

Textbook

Serway, *Physics for Scientists and Engineers*, 5th Edition, Saunders, 1999.

Goals

An understanding of the fundamentals of mechanics, and competence in solving problems at the level of elementary calculus.

Prerequisite by Topic

Calculus – derivatives, antiderivatives
Sequences and series
Limits

Topics

Measurements, units, and vectors (3 hours)
Particle kinematics (motion) (6 hours)
Particle dynamics (forces) (6 hours)
Work and energy (6 hours)
Systems of particles (momentum and collisions) (3 hours)
Rotational kinematics (3 hours)
Rotational dynamics and statics (3 hours)

COURSE DESCRIPTION

Department and Course Number: PHY 131L

Course Coordinator: George Rainey, Professor of Physics

Course Title: General Physics Laboratory

Total Credits: 1

Current Catalog Description

Laboratory to accompany General Physics lecture series. Experiments in mechanics, hydrostatics, wave, thermodynamics, optics, and electricity and magnetism. 1 three-hour laboratory. To be taken concurrently with PHY 131.

Textbook

Cal Poly Physics Department. *Physics 131 Laboratory Manual*.

Goals

To gain an understanding of fundamental principles in mechanics through the direct experience of laboratory investigation; and to develop skills in preparation of a critical, concise written summary of experimental results.

Prerequisite by Topic

Calculus – derivatives
Sequences and series
Limits

Topics

Vector arithmetic (force table) (3 hours)
Empirical equations (deflection of a beam) (3 hours)
Propagation of uncertainty (determination of density) (3 hours)
Simple pendulum (3 hours)
Atwood machine (3 hours)
Uniform circular motion (3 hours)
Ballistic pendulum/projectile motion (3 hours)
Archimedes' Principle (specific gravity) (3 hours)

COURSE DESCRIPTION

Department and Course Number: PHY 132

Course Coordinator: George Rainey, Professor of Physics

Course Title: General Physics

Total Credits: 3

Current Catalog Description

Fundamental principles of harmonic motion, gravity, fluid mechanics, waves, thermodynamics, kinetic theory, and optics. 3 lecture/problems. Prerequisites: C- or better in PHY 131, PHY 131L. Corequisites: MAT 116 and PHY 132L.

Textbook

Serway, *Physics for Scientists and Engineers*, 5th Edition, Saunders, 1999.

Goals

An understanding of the fundamentals of rotational dynamics, mechanical waves and vibrations, thermodynamics, and optics; and competence in solving problems at the level of elementary calculus.

Prerequisite by Topic

Calculus – derivatives, antiderivatives
Sequences and series
Limits
Measurement, units, vectors
Work and energy

Topics

Oscillations (4 hours)
Gravitation (2 hours)
Fluid mechanics (3 hours)
Mechanical waves (3 hours)
Sound (3 hours)
Temperature (3 hours)
Heat and First Law of Thermodynamics (3 hours)
Kinetic theory of gases (3 hours)
Entropy and Second Law of Thermodynamics (3 hours)
Nature of light (2 hours)
Optics (1 hour)

COURSE DESCRIPTION

Department and Course Number: PHY 132L

Course Coordinator: George Rainey, Professor of Physics

Course Title: General Physics Laboratory

Total Credits: 1

Current Catalog Description

Laboratory to accompany General Physics lecture series. Experiments in mechanics, hydrostatics, wave, thermodynamics, optics, and electricity and magnetism. 1 three-hour laboratory. To be taken concurrently with PHY 132.

Textbook

Cal Poly Physics Department. *Physics 132 Laboratory Manual*.

Goals

To gain an understanding of fundamental principles in the mechanics of fluids and vibrating systems, thermodynamics, and optics through the direct experience of laboratory investigation; and to develop skills in preparation of a critical, concise written summary of experimental results.

Prerequisite by Topic

Calculus – derivatives
Sequences and series
Limits
Measurements, units, and vectors
Work and energy

Topics

Rotational dynamics (3 hours)
Simple harmonic motion (3 hours)
Standing waves (3 hours)
Resonance of air columns (3 hours)
Thin lenses (3 hours)
Equation of state for an ideal gas (3 hours)
Coefficient of linear thermal expansion (3 hours)
Calorimetry (specific heat) (3 hours)

COURSE DESCRIPTION

Department and Course Number: PHY 133

Course Coordinator: George Rainey, Professor Physics

Course Title: General Physics

Total Credits: 3

Current Catalog Description

Fundamental principles of electricity and magnetism. Coulomb's law, electric fields, potential properties of dielectrics, capacitance, Ohm's law, magnetism and magnetic fields, measuring instruments, magnetic field of moving charges, induced emf. 3 lecture/problems. Prerequisites: C- or better in PHY 131, PHY 131L and MAT 116. Corequisite: PHY 133L.

Textbook

Serway, *Physics for Scientists and Engineers*, 5th Edition, Saunders, 1999.

Goals

An understanding of the fundamentals of electricity and magnetism, and competence in solving problems at the level of elementary calculus.

Prerequisite by Topic

Calculus - derivatives
Sequences and series
Limits
Measurement and units
Work and energy

Topics

Electric charge and electric field (4 hours)
Electric potential and capacitance (4 hours)
Electric current and resistance (3 hours)
Electromotive force and DC circuits (4 hours)
Magnetic field, sources and effects (5 hours)
Electromagnetic induction and inductance (4 hours)
Magnetic properties of matter (3 hours)
Electromagnetic oscillations (3 hours)

COURSE DESCRIPTION

Department and Course Number: PHY 133L

Course Coordinator: George Rainey, Professor of Physics

Course Title: General Physics Laboratory

Total Credits: 1

Current Catalog Description

Laboratory to accompany General Physics lecture series. Experiments in mechanics, hydrostatics, wave, thermodynamics, optics, and electricity and magnetism. 1 three-hour laboratory. To be taken concurrently with PHY 133.

Textbook

Cal Poly Physics Department. *Physics 133 Laboratory Manual*.

Goals

To gain an understanding of fundamental principles in electricity and magnetism through the direct experience of laboratory investigation; and to develop skills in preparation of a critical, concise written summary of experimental results.

Prerequisite by Topic

Calculus - derivatives
Sequences and series
Limits
Measurements, units, and vectors
Work and energy

Topics

Electrolysis (3 hours)
Electric field plotting (3 hours)
Response of circuit elements (3 hours)
Analog galvanometers (3 hours)
Electron charge-to-mass ratio (3 hours)
Current balance; magnetic field balance (3 hours)
Oscilloscope (3 hours)
Magnetic field (3 hours)

COURSE DESCRIPTION

Department and Course Number: BIO 110

Course Coordinator: Ronald Quinn, Professor of Biological Sciences

Course Title: Life Science

Total Credits: 3

Current Catalog Description

Basic concepts in the study of living systems, including human beings. The study of biology will be used to illustrate approaches of science in understanding the universe. The role of science in modern society and the impact of human civilization on other organisms will be considered. Designed to satisfy the general education requirements for life science. 3 lecture-problems.

Textbook

Brum, Gilbert and McKane, Larry. *Biology: Exploring Life*, John Wiley & Sons, 1989.

Goals

Students will gain a broad understanding of the nature of living systems, the diversity of living forms, evolutionary processes, modern genetics, the importance of natural processes to human welfare, and the impact of human activities on nature. Students will gain the perspective necessary to make informed decisions about contemporary societal issues involving life sciences such as: public policy and AIDS; teaching of evolution vs. creationism in public schools; abortion; environmental concerns; genetic engineering; and bioethics. Students will gain an appreciation of the role science and the scientific method have played in the development of the technological society in which we live.

Prerequisite by Topic

There are no prerequisites for this course.

Topics

Scientific method (3 hours)
Chemistry (3 hours)
Cells (3 hours)
Energy (3 hours)
Physiological systems (3 hours)
Genetics (3 hours)
Reproduction (3 hours)
Evolution (3 hours)
Biodiversity (3 hours)
Ecology (3 hours)

COURSE DESCRIPTION

Department and Course Number: CHM 121, 122, 123 General Chemistry

Course Title: General Chemistry

Total Credits: 3 (for CHM 121 alone)

Current Catalog Description

Atomic theory of structure and bonding, chemical equations, gas laws, oxidation-reduction, electrochemistry, states of matter, equilibrium, acids and bases, thermodynamics and reaction kinetics and their applications to chemistry, physics, and engineering sciences. For majors requiring calculus. 3 lectures/problem solving. Prerequisite to CHM 121: high school chemistry or CHM 103/103A and high school algebra. Concurrent: CHM 121L, 122L, 123L respectively

Textbook

General Chemistry, Fifth Edition, Ebbing

Goals

To provide students with an appreciation of chemistry, both theoretically and in the laboratory.

Prerequisites by Topic

High school chemistry or equivalent
Two years of algebra

Topics

Atomic structure
Balancing of equations
Chemical reactions
Stoichiometry
Mole concepts
Gas laws
Bohr theory
Quantum Numbers
Resonance
Bonds

Laboratory Projects

Density determination
Synthesis of Alum
Salt solubilities
Analysis of a hydrated metal sulfate
Redox reaction
Spectrophotometric analysis
Molar volume of oxygen