

## **COURSE DESCRIPTION**

Department and Course Number	CS 112	Course Coordinator	Gyorgy Petruska
------------------------------	--------	--------------------	-----------------

Course Title	Survey of Computer Science	Total Credits	3.0
--------------	----------------------------	---------------	-----

### **Current Catalog Description**

This course is designed to provide a broad and realistic idea of what computer professionals do and how they do it. It will prepare students for other computing courses, including software development courses, by providing both individual and team hands-on lab experiences with Web design, markup languages (HTML) and JavaScript. Students will be introduced to various professional opportunities and work environments. Current topic in computer science as they relate to society will be covered. Students will gain sufficient programming experience to enable smooth transition to CS 160 Java programming.

### **Updated Catalog Description (2009 – 2010)**

The objective of this course is an understanding of technology covering a wide range of topics including hardware and software, architecture, networking and the Internet, file structures, security, databases, file systems, and an introduction to programming concepts including writing code in HTML, JAVA, and JavaScript while weighing the social and ethical impact of technology on society.

### **Textbooks**

- Connecting with Computer Science, Anderson, Ferro, Hilton. Thomson Learning, 2005
- HTML and JavaScript Basics, 3<sup>rd</sup> Edition, Karl Barksdale & E. Shane Turner, Thomson Learning, 2005

### **References**

None

### **Course Learning Outcomes**

Upon successful completion of the course requirements, a student should be able to:

1. Identify the different parts of a computer and explain what they do.( a,b,c,ef,g,h,i)
2. Outline basic computer encryption and understand how it works.( a,b,c,e,f,g,h,i)
3. Identify common file systems and explain the pros and cons.( a,b,c,e,f,g,h,i)
4. Identify and label the common components of a database system.( a,b,c,e,f,g,h,i)
5. Use basic JAVA Programming to enhance my web pages.( a,c,d)
6. Write basic HTML based web pages.( a,c,d)
7. Explain the difference between HTML and JAVA Coding.( a,c,d)
8. Understand the ethical issues surrounding intellectual property and software.  
(e,f,g)
9. Identify and explain network protocol packets as they travel across a network.  
( a,b,c,i)
10. Can explain the difference between binary and ascii encoding.( a,b,c,i)
11. Can explain how the Internet exchanges, stores, and moves information across different systems.( a,c,e,f,g,h,i)

### **Prerequisites by Topic**

None.

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•	•	•		•	•	•	•	•		
2	•	•	•		•	•	•	•	•		
3	•	•	•		•	•	•	•	•		
4	•	•	•		•	•	•	•	•		
5	•		•	•							
6	•		•	•							
7	•		•	•							
8					•	•	•				
9	•	•	•						•		
10	•	•	•						•		
11	•		•	•		•	•	•	•		

### Program Outcomes

- a. An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- c. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- d. An ability to function effectively on teams to accomplish a common goal.
- e. An understanding of professional, ethical, legal, security and social issues and responsibilities.
- f. An ability to communicate effectively with a range of audiences.
- g. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- h. Recognition of the need for and an ability to engage in continuing professional development.
- i. An ability to use current techniques, skills, and tools necessary for computing practice.
- j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- k. An ability to apply design and development principles in the construction of software systems of varying complexity.

### Major Topics Covered in the Course (Academic hours)

A Brief History of Computing	1
Software Tools for Techies	3
Computer Architecture	3
Numbering Systems	2
Operating Systems	2
Networks	3
The Internet	2
HTML Coding	3
JavaScript Coding	3

Databases	3
Data Structures	3
File Structures	3
Programming	3
Software Engineering	3
Computers Ethics and Security	2
Emerging Technologies	1
Two Exams	3

### **Laboratory projects**

(All are 1 week assignments except Lab 0)

0. Editing, saving, and execution of HTML code using Windows Notepad.
1. Construction/completion of small HTML web applications that manipulate text and graphics.
2. Construction/completion of small HTML web applications that involve input/output with forms.
3. Construction/completion of small JavaScript web applications that manipulate text and graphics.
4. Construction/completion of small JavaScript web applications that use functions, timers, and variables.
5. Construction/completion of small JavaScript web applications that use arrays and conditioners to manipulate graphical images.
6. Construction/completion of small JavaScript web applications that use error checking against input/output forms.

### **Project assignments**

- a. A fully integrated HTML and JavaScript web application that combines everything that we have covered in class. The student is allowed to present the code in any way they wish as long as they have all of the requirements of the project. (8 weeks)

### **Assessment Plan for the Course**

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

### **Credit Content**

The course requires 3 credit hours for lectures per week (1 credit or academic hour is 50 minutes), making 45 hour per semester including the 3 hours allocated for 2 midterms. The laboratory projects require 11/2 credit hours per week, adding up to 15 credit hours per semester, altogether 60 credit hour per semester. Final exam not included.

### **Estimated Curriculum Category Content**

Version 1 Based upon the 3 credit hours for lectures, laboratory not included

Minimum of 0.5 credit hours. The 3 credit-hour lecture contingent has 45 contact hours. CS161 uses 3 of these for exams. 0.5 credit hours is equivalent to  $0.5/3 \times 42$  or 7 contact hours.

	COR E	ADVANC ED		COR E	ADVANC ED
Data Structures	0.5	0	Computer Organization and Architecture	2.0	0
Algorithms	0	0	Concepts of Programming Languages	0.5	0
Software Design	0	0			

**Written reports**

None.

**Social and Ethical Issues**

Computer Ethics and Security.

## COURSE DESCRIPTION

Department and Course Number : CS 114  
Semester Hours: 3.0

Course Title: Introduction to Visual Basic  
Course Coordinator: Robert J. Sanders

### Current Catalog Description

#### **C: MA 150 or MA 153**

This course provides an introduction to programming using the Visual Basic language and its integrated development environment. Topics to be covered include the syntax and structure of the VB language; controls, dialog boxes, and other interface tools; menu design; multiple forms; error-trapping; and arrays. Other topics that may be covered include object linking and embedding (OLE); VB for applications; database development using record sets and databound controls; data handling; grids; validation and election; drag and drop; and graphics; and new revisions for interoperability with other languages.

#### **Textbook**

##### PROGRAMMING IN VISUAL BASIC 2008

J. C. BRADLEY AND ANITA C. MILLSPAUGH

United States, Irwin/McGraw-Hill, 2008 (ISBN 978-0-07-351720-9 / MHID 0-07-351720-8)

#### **References**

None.

#### **Course Outcomes**

A student who successfully fulfills the course requirements will have demonstrated an ability to:

The letters in parentheses refer to ABET Program Learning Outcomes. \*\*

1. Describe the process of visual program design and development using the Microsoft Visual Studio integrated environment (IDE) - (b,c,i,k) \*\*
2. Explain and use object-oriented programming - (a,b,c) \*\*
3. Design a user interface (GUI) - (b,c,k) \*\*
4. Distinguish between variables and constants and use them in calculations - (a,b,c,i,j,k) \*\*
5. Evaluate decisions and conditions using control structures (If-Then-Else, Loops, etc.) - (b,c,i,j,k) \*\*
6. Create menus and submenus for program control - (a,b,c,i) \*\*
8. Use multiple forms in a project - (b,c,k) \*\*
9. Develop lists and print reports - (b,c) \*\*
10. Establish an array and reference individual elements using subscripts - (b,i) \*\*
11. Program using the Visual Web Developer - (b,i,k) \*\*
12. Access database files - (b,c) \*\*
13. Save data in a data file - (b,c) \*\*
14. Use Graphics methods to draw shapes, lines and filled shapes - (b,c) \*\*
15. Display a Web page on a Windows form using a WebBrowser control - (b,c,k) \*\*

## Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1		•	•						•		•
2	•	•	•								
3		•	•								•
4	•	•	•						•	•	•
5		•	•						•	•	•
6	•	•	•						•	•	
7		•	•						•		•
8		•	•								•
9		•	•								
10		•							•		
11		•							•		•
12		•	•								
13		•	•								
14		•	•								
15		•	•								•

## Program Outcomes

- a An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- b An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- c An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- d An ability to function effectively on teams to accomplish a common goal.
- e An understanding of professional, ethical, legal, security and social issues and responsibilities.
- f An ability to communicate effectively with a range of audiences.
- g An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- h Recognition of the need for and an ability to engage in continuing professional development.
- i An ability to use current techniques, skills, and tools necessary for computing practice.
- j An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- k An ability to apply design and development principles in the construction of software systems of varying complexity.

## Prerequisites by Topic

A basic course in Algebra and Trigonometry is required.

### Major Topics Covered in the Course (class hours)

1. Visual program design and development using the Microsoft Visual Studio integrated environment (IDE) - (1.25 Hour)
2. Object-oriented programming - (1.25 Hour)
3. Design a user interface (GUI) - (1.25 Hour)
4. Distinguish between variables and constants and use them in calculations - (1.25 Hour)
5. Evaluate decisions and conditions using control structures (If-Then-Else, Loops, etc.) - (1.25 Hour)
6. Create menus and submenus for program control - (1.25 Hour)
7. Write reusable code in sub procedures and function procedures and call procedures from other locations - (1.25 Hour)
8. Use multiple forms in a project - (1.25 Hour)
9. Develop lists and print reports - (1.25 Hour)
10. Establish an array and reference individual elements using subscripts - (1.25 Hour)
11. Program using the Visual Web Developer - (1.25 Hour)
12. Access database files - (1.25 Hour)
13. Save data in a data file - (1.25 Hour)
14. Use Graphics methods to draw shapes, lines and filled shapes - (1.25 Hour)
15. Display a Web page on a Windows form using a WebBrowser control - (1.25 Hour)
16. Develop eight projects covering the above topics – (10 Hours)
17. Take three examinations over the above topics – (4.5 Hours)

33.25 Hours

### Assessment Plan for the Course

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: “Strongly agree”, “Agree”, “Neither agree nor disagree”, “Disagree”, and “Strongly Disagree”. Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

### How Data in the Course is Used to Assess Program Outcomes

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see “Relationship between Course Outcomes and Program Outcomes” above). For CS 114, Program Outcomes a, b, c, i, j, and k are supported.

### Estimate Curriculum Category Content

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0	0	Software design	0.5	0
Data structures	0	0	Concepts of programming languages	2.5	0
Computer organization and architecture	0	0		0	0

## COURSE DESCRIPTION

Department and Course Number                      CS 160                      Course Coordinator      Gyorgy Petruska

Course Title      Introduction to Computer Science I                      Total Credits      4.0

### Current Catalog Description

An introduction to computer concepts and the fundamentals of structured programming in a high-level language. Problem-solving techniques, specifications, stepwise refinement, programming style, structure charts, and program documentation. Programming topics include data types, assignments, input/output, subprograms, selection, iteration, arrays, records, text files, and simple searching and sorting. P: CS 112 or equivalent experience and MA 153.

### Updated Catalog Description (2009 – 2010)

The course is an introduction to the fundamental concepts and techniques of Computer Science. Students will learn to program using an object-oriented language. They will learn how to translate a real problem into a program description, and how to write and test a program to implement their description. The emphasis will be on developing a professional style at an elementary level. CS 160 will carry syntax as far as interacting classes, arrays of one dimension, and simple file i/o. Students with no programming background should instead consider CS 112. (Prerequisite MA 153)

### Textbook

Starting Out with Java; from Control Structures through Objects 4<sup>th</sup> ed. by Tont Gaddis, Addison Wesley, New York, N.Y., 2009 (Required)

### References

None

### Course Learning Outcomes

Upon successful completion of the course requirements, a student should be able to:

- Design simple computer programs in the Java language (b,c)
- Understand the main organizational principles of Java programs (c,i)
- Translate a problem description into a program design (b,i)
- Understand the details of Java (i)
- Apply variable types (i)
- Apply control structures in Java programs (i)
- Implement input/output code for file, the console or GUI (b,c,i)
- Know how to declare data and define methods in a Java class (c,i)
- Understand class interaction in Java (c,i)
- Know how to apply and manipulate one dimensional arrays (i)
- Know how to find the errors in the program (i)
- Know how to write temporary modifications in the program to find errors (i)
- Interpret error messages of the computer (i)

### Prerequisites by Topic

None.

**Relationship between Course Outcomes and Program Outcomes**

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1		•	•								
1a			•						•		
1b		•							•		
2									•		
2a									•		
2b									•		
2c		•	•						•		
2d									•		
2e			•						•		
2f									•		
3									•		
3a									•		
3b									•		

**Program Outcomes**

- a. An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- c. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- d. An ability to function effectively on teams to accomplish a common goal.
- e. An understanding of professional, ethical, legal, security and social issues and responsibilities.
- f. An ability to communicate effectively with a range of audiences.
- g. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- h. Recognition of the need for and an ability to engage in continuing professional development.
- i. An ability to use current techniques, skills, and tools necessary for computing practice.
- j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- k. An ability to apply design and development principles in the construction of software systems of varying complexity.

**Major Topics Covered in the Course (Academic hours)**

- Computer systems and programming languages..... (3)
- Java fundamentals..... (6)
- Decision structures..... (6)
- Iteration Structures..... (4)
- File input/output..... (3)
- Application and properties of methods in Java..... (6)
- Classes, class members, constructors..... (4)
- One dimensional arrays..... (4)
- Classes, objects, aggregation, object oriented design..... (6)
- Mid-semester examinations..... (3)

## **Laboratory projects**

(All are 1 week assignments except Lab 0)

0. Editing, compiling, and execution Java applications in the Eclipse environment
1. Construction/completion of small Java applications involving simple input/output, arithmetic expression evaluation. A short report attached to the program as a comment is required
2. Construction/completion of small Java applications involving various data types and String methods. A short report attached to the program as a comment is required
3. Construction/completion of small Java applications involving Scanner class and JOptionPane class methods
4. Construction/completion of small Java applications involving if statements utilizing relational and Boolean operators
5. Construction/completion of small Java applications involving loops and file i/o
6. Construction/completion of a Java class containing methods of various properties
7. Construction/completion of two Java classes involving constructors, accessor and mutator methods, aggregation
8. Construction/completion of three Java classes involving collaboration, copy constructors, array manipulation

## **Project assignments**

- a. Given a specification, construct a Java application that simulates a simplified arcade game based on the equations of projectile motion. The solution involves utilization of library classes for input/output, evaluating trigonometric functions. The solution also involves declaring and operating on variable and constant numeric and character data, testing. (4 weeks)
- b. Given a specification, construct a Java application that simulates the activities of money exchange in a bank. The solution involves the construction of a Java class having several fields and methods, selection and iteration structures, GUI and file input/output, testing. (4 weeks)
- c. Given a specification, construct a Java application that simulates the daily traffic of a parking garage. The solution involves implementing a class hierarchy using four classes, UML diagram interpretation, random selection, array manipulation, testing. (4 weeks)

## **Assessment Plan for the Course**

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

## **Credit Content**

The course requires 3 credit hours for lectures per week (1 credit or academic hour is 50 minutes), making 45 hour per semester including the 3 hours allocated for 2 midterms. The laboratory projects require 11/2 credit hours per week, adding up to 15 credit hours per semester, altogether 60 credit hour per semester. Final exam not included.

## Estimated Curriculum Category Content

Version 1 Based upon the full credit content of the course laboratory included

Minimum of 0.5 credit hours. A 4 credit-hour class has 60 contact hours. CS161 uses 3 of these for exams.

0.5 credit hours is equivalent to  $0.5/4*57$  or about 7 contact hours.

	CORE	ADVANCED		CORE	ADVANCED
Data Structures	0.5	0	Computer Organization and Architecture	0	0
Algorithms	0.5	0	Concepts of Programming Languages	2.0	0
Software Design	1.0	0			

Version 2 Based upon the 3 credit hours for lectures, laboratory not included

Minimum of 0.5 credit hours. The 3 credit-hour lecture contingent has 45 contact hours. CS161 uses 3 of these for exams. 0.5 credit hours is equivalent to  $0.5/3*42$  or 7 contact hours.

	CORE	ADVANCED		CORE	ADVANCED
Data Structures	0.5	0	Computer Organization and Architecture	0	0
Algorithms	0.5	0	Concepts of Programming Languages	1.5	0
Software Design	0.5	0			

### Written reports

See Lab 1 and Lab 2

### Social and Ethical Issues

None

## **COURSE DESCRIPTION**

Department and Course Number                      CS 160H                      Course Coordinator    Robert Sedlmeyer

Course Title    Introduction to Computer Science I Honors                      Total Credits    4.0

### **Current Catalog Description**

An introduction to Java programming features with emphasis on an object-oriented approach. P: honors eligibility with an SAT math score of 600 or higher, placement into MA 165, or consent of instructor, Equivalent of CS 160 for honors students.

## COURSE DESCRIPTION

Department and Course Number                      CS 161                      Course Coordinator    Gyorgy Petruska

Course Title    Introduction to Computer Science II                      Total Credits    4.0

### Current Catalog Description

Continuation of CS 160. Emphasis is put on program development including programming style, modularization, data abstraction, abstract data types, and selection and analysis of algorithms; programming using a structured approach. Topics include recursion, pointers, files, and elementary data structures including stacks, queues, linked lists, and binary trees.

### Updated Catalog Description

This course continues CS 160. Students will design larger programs to solve more complicated problems. The emphasis is on deepening students' abilities to deal with abstraction, problem decomposition, and the interaction between program components. Students will develop their professional practice through analysis of more general problems, debugging and testing of their programs, and written presentation of their solutions. Topics include multidimensional arrays, event-driven programs, GUI's, class inheritance and interfaces, and libraries. (Prerequisite 160 and co-requisite MA 175)

### Textbook

Starting Out with Java; from Control Structures through Objects 4<sup>th</sup> ed. by Tont Gaddis, Addison Wesley, New York, N.Y., 2009 (Required)

### References

None

### Prerequisites by Topic

CS 160.

### Course Learning Outcomes

Upon successful completion of the course requirements, a student should be able to:

1. Recognize the need for arrays in programming tasks, and I can manipulate data in one and multi-dimensional arrays (b, c, i, j)
2. Apply basic algorithms for sorting and searching arrays (i, j)
3. Work with ArrayList objects (i, j)
4. Apply recursive programming a, b, c, i)
5. Apply and utilize fundamental GUI (c, h, i)
6. Design and implement multi-class solutions, in particular apply inheritance (b, c, i)
7. Create exception classes and handle Java exceptions (b, c, i)
8. Utilize event driven programming and interfaces in advanced GUI applications (b, c, i)
9. Recognize and apply Model-View-Controller architecture (b, c, i)

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1		•	•						•	•	
2									•	•	
3									•	•	
4	•	•	•						•		
5			•					•	•		
6		•	•						•		
7		•	•						•		
8		•	•						•		
9		•	•						•		

### Program Outcomes

- a. An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- c. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- d. An ability to function effectively on teams to accomplish a common goal.
- e. An understanding of professional, ethical, legal, security and social issues and responsibilities.
- f. An ability to communicate effectively with a range of audiences.
- g. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- h. Recognition of the need for and an ability to engage in continuing professional development.
- i. An ability to use current techniques, skills, and tools necessary for computing practice.
- j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- k. An ability to apply design and development principles in the construction of software systems of varying complexity.

### Major Topics Covered in the Course (Academic hours)

Basic GUI constructions	(6)
Multi-dimensional arrays and ArrayList objects	(4)
Text processing and wrapper classes	(6)
Inheritance and polymorphism	(6)
Exception handling and advanced file i/o	(5)
Advanced GUI applications	(6)
Applets	(3)
MVC architecture	(3)
Recursive programming	(3)
Mid-semester exams	(3)

### Laboratory projects

(All are 1 week assignments)

1. Construction/completion of small Java applications involving a simple GUI window implementing it in two versions, extending JFrame class first, aggregating a JFrame object second.

2. A continuation of Exercise 1 with adding button functionalities, applying JPanel objects to follow the required layout.
3. Construction/completion of small Java applications involving two-dimensional arrays representing numerical matrices.
4. Construction/completion of small Java applications involving array-based lists, practice unsorted and sorted list operations
5. Construction/completion of a Java applications involving three classes forming an inheritance hierarchy and an application class. The solution requires understanding the problem description and creating and implementing the student's own design.
6. This assignment requires the design and implementation of two small Java applications. In the first students gain experience with library exception classes, and with handling the thrown exceptions. In the second students are required to create their own exception classes providing more robust solution to the problem.
7. Construction/completion of small Java project involving an advanced GUI application with JSlider objects, exception handling and GUI related interfaces.
8. Construction/completion of a Java applet involving a clock face and timer driven events. Extra credit offered for a more complete and realistic clock with additional functions and with the HTML code associated to the applet.

### **Project assignments**

1. Given a specification, construct a Java GUI application that simulates a simplified casino game. The user puts a bet on one of two colors which is matched against a color randomly selected by the computer from three colors. The program displays winnings and losses. The solution involves object oriented multi-class design, user defined GUI, event driven programming, coloring and bordering GUI components, using disabled and enabled components. (4 weeks)
2. Given a specification, construct a Java application that simulates the banking activities concerning customer accounts, applying various GUI windows for display. The solution involves object oriented multiclass design, inheritance, exception handling, manipulation of ArrayList objects and parallel arrays, observing the MVC paradigm, class collaboration. (4 weeks)
3. Given a specification, construct a Java application that simulates a random race, where at each stride the contenders randomly make a forward or backward move on the track. The user selects the track size, the contenders. The solution involves object oriented multiclass design, application of the MVC architecture, ArrayList objects, advanced GUI constructs, inner classes, timer events. (4 weeks)

### **Assessment Plan for the Course**

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

**Credit Content**

The course requires 3 credit hours for lectures per week (1 credit or academic hour is 50 minutes), making 45 hour per semester including the 3 hours allocated for 2 midterms. The laboratory projects require 11/2 credit hours per week, adding up to 15 credit hours per semester, altogether 60 credit hour per semester. Final exam not included.

**Estimate Curriculum Category Content**

Version 1 Based upon the full credit content of the course laboratory included  
 Minimum of 0.5 credit hours. A 4 credit-hour class has 60 contact hours. CS161 uses 3 of these for exams. 0.5 credit hours is equivalent to 0.5/4\*57 or about 7 contact hours.

	COR E	ADVANC ED		COR E	ADVANC ED
Data Structures	0.5	0	Computer Organization and Architecture	0	0
Algorithms	0.5	0	Concepts of Programming Languages	2.0	0
Software Design	1.0	0			

Version 2 Based upon the 3 credit hours for lectures, laboratory not included  
 Minimum of 0.5 credit hours. The 3 credit-hour lecture contingent has 45 contact hours. CS161 uses 3 of these for exams. 0.5 credit hours is equivalent to 0.5/3\*42 or 7 contact hours.

	COR E	ADVANC ED		COR E	ADVANC ED
Data Structures	0.5	0	Computer Organization and Architecture	0	0
Algorithms	0.5	0	Concepts of Programming Languages	1.5	0
Software Design	0.5	0			

**Oral and Written Communications**

Written reports are part of some laboratory assignments

**Social and Ethical Issues**

None.

## COURSE DESCRIPTION

Department and Course Number : CS 203  
Semester Hours: 3.0

Course Title: Advanced Visual Basic  
Course Coordinator: Robert Sedlmeyer

### Current Catalog Description

P: CS 114 or ECET 114

This course continues the study of Visual Basic begun in CS 114/ECET 114. Topics to be covered include reading and writing of sequential and direct files; custom controls; advanced SQL; the creation of online help; object linking and embedding (OLE); calling DLL procedures (Windows API); class modules ; and an introduction to ActiveX components. Student will learn the skills needed to create stand-alone and www-based Visual Basic applications for personal computer use. The course will provide guidance in preparing for the Microsoft Certified Systems Designer examination.

### Textbook

ADVANCED PROGRAMMING USING VISUAL BASIC.NET 2005

J. C. BRADLEY AND ANITA C. MILLSPAUGH

United States, Irwin/McGraw-Hill, 2008 (ISBN 978-0-07-351717-9 / MHID 0-07-351717-8)

### References

None.

### Course Outcomes

A student who successfully fulfills the course requirements will have demonstrated an ability to:

The letters in parentheses refer to ABET Program Learning Outcomes. \*\*

1. Describe the process of visual program design in developing an MDI Document interface in using the .NET Framework in VS.net – (a, b, c, i, k) \*\*
2. Explain how to build a multitier program using classes - (a,b,c, i, k) \*\*
3. Design a Windows database application - (b,c,k) \*\*
4. Distinguish between Windows database and a Web database using related - (a,b,c,i,j,k) \*\*
5. Evaluate how to use a data grid to update a Windows database table - (b,c,i,j,k) \*\*
6. Create and consume Web Services - (a,b,c,i) \*\*
7. Write applications using Web Forms with ASP.NET - (b,c,i,k) \*\*
8. Use a Web Form database and maintain state of data - (b,c,k) \*\*
9. Develop database reports using Crystal Reports - (b,c) \*\*
10. Establish and use a collection of objects - (b,i) \*\*
11. Create and use user controls - (b,i,k) \*\*
12. Create your own Help Files using the HTML help Workshop - (b,c) \*\*
13. Using emulators to develop applications for mobile and smart devices - (b,c) \*\*

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1		•	•						•		•
2	•	•	•								
3		•	•								•
4	•	•	•						•	•	•
5		•	•						•	•	•
6	•	•	•						•	•	
7		•	•						•		•
8		•	•								•
9		•	•								
10		•							•		
11		•							•		•
12		•	•								
13		•	•								

### Program Outcomes

1. An ability to apply knowledge of computing and mathematics appropriate to the discipline.
2. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
3. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
4. An ability to function effectively on teams to accomplish a common goal.
5. An understanding of professional, ethical, legal, security and social issues and responsibilities.
6. An ability to communicate effectively with a range of audiences.
7. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
8. Recognition of the need for and an ability to engage in continuing professional development.
9. An ability to use current techniques, skills, and tools necessary for computing practice.
10. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
11. An ability to apply design and development principles in the construction of software systems of varying complexity.

### Prerequisites by Topic

A basic course in CS 114 Introduction to Visual Basic and a course in Algebra and Trigonometry are required.

#### Major Topics Covered in the Course (class hours)

1. Introduction, review and Visual Studio.NET (5.0)
2. Building Multitier with Classes (3.0)
3. Windows Database Applications, related Tables and Updates) - (7.5 Hour)
4. Using Web Services, Web Forms – ASP.NET (4.5)
5. Web Forms Database (3.0)
6. Writing Database Reports – Crystal Report (3.0)
7. Using Collections (3.0)
8. User Controls (1.5)
9. Create Help Files (3.0)
10. Looking Ahead (3.0)

### **Assessment Plan for the Course**

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: “Strongly agree”, “Agree”, “Neither agree nor disagree”, “Disagree”, and “Strongly Disagree”. Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

### **How Data in the Course is Used to Assess Program Outcomes**

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see “Relationship between Course Outcomes and Program Outcomes” above). For CS 203, Program Outcomes a, b, c, i, j, and k are supported.

### **Estimate Curriculum Category Content**

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0	0	Software design	0.5	0
Data structures	0	0	Concepts of programming languages	2.5	0
Computer organization and architecture	0	0		0	0

### **Oral and Written Communications**

Written reports are part of some laboratory assignments

### **Social and Ethical Issues**

None.

## COURSE DESCRIPTION

Department and Course Number: CS 232  
Semester Hours: 3.0

Course Title: Introduction to C & Unix  
Course Coordinator: Britton Wolfe

### Current Catalog Description

This course is an introduction to the C language and the Unix operating system. It presumes fluency in a high-level language. The course will focus on standard C and Unix tools, rather than a proprietary version of either. C topics include data types, the syntax for arithmetic, logical and relational functions, control functions, scope, communications with the shell, file i/o, pointers, arrays, structs, typedefs, macro and preprocessor functions, and the use of libraries and multiple source files. Unix topics include the file and directory structures, permissions, shells, standard tools such as history, sort, vi, grep, sed, tar, and make, and simple shell scripting.

### Textbook

Brian W. Kernighan & Dennis M. Richie, *The C Programming Language*, 2<sup>nd</sup> ed. Prentice-Hall, 1988. ISBN 0-13-110370-9 (hardback) or 0-13-110362-8 (paperback).

### References

Michael Loukides, & Andrew Oram, *Programming with GNU Software*. O'Reilly, 1996. ISBN 1-56592-112-7.

### Course Outcomes

Upon successful completion of the course requirements, a student should be able to:

1. Demonstrate a working knowledge of the C programming language, including operators, functions, structures, arrays, and pointers. (i)
2. Use standard C libraries to read and write files in several different formats. (b, c, i)
3. Demonstrate familiarity with the Unix/Linux shell and file system. (i)
4. Write, debug, and edit C programs using standard Unix/Linux tools. (c, i)
5. Compile and link programs with multiple source files and precompiled libraries. (i)
6. Work with a team to design and implement software modules that require the consideration of computational tradeoffs. (a, b, c, d, i, j)
7. Write software that is not vulnerable to “buffer overflow” attacks, and understand how such attacks can compromise a system. (e, i)
8. Use standard Unix/Linux shell tools and scripts to automate simple tasks over multiple files. (i)

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1									•		
2		•	•						•		
3									•		
4			•						•		
5									•		
6	•	•	•	•					•	•	
7					•				•		
8									•		

### Program Outcomes

- a. An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- c. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- d. An ability to function effectively on teams to accomplish a common goal.
- e. An understanding of professional, ethical, legal, security and social issues and responsibilities.
- f. An ability to communicate effectively with a range of audiences.
- g. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- h. Recognition of the need for and an ability to engage in continuing professional development.
- i. An ability to use current techniques, skills, and tools necessary for computing practice.
- j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- k. An ability to apply design and development principles in the construction of software systems of varying complexity.

### Prerequisites by Topic

A two-course sequence in high-level language programming is assumed to have been completed before this course.

### Major Topics Covered in the Course (class hours)

1. Unix/Linux shell and file system (2)
2. Basic development tools (1)
3. Variables, operators, and expressions (3)
4. Control flow statements (1)
5. Functions (2)
6. Pointers and arrays (5)
7. Structures (3)
8. File I/O (2)
9. Standard library functions (3)
10. Program structure (2)
11. Compiling and linking programs with multiple files and/or libraries (1)
12. Using Makefiles (2)
13. System calls (1)
14. UNIX/Linux tools (6)
15. Shell scripting (5)
16. Exams and review (6)

## Assessment Plan for the Course

There are several assessment tools in place to measure how well the Course Outcomes are being achieved. Each of these is listed below, with references to the particular Course Outcomes that will be measured by that assessment tool.

1. Programming projects (1,2,4,5,6,7)
2. Exams (1,2,3,4,5,7,8)
3. Homework/Quizzes (1,2,3,4,5,7,8)
4. Peer evaluation (6)
5. Self-assessment (1,2,3,4,5,6,7,8)

The following table illustrates that every Course Outcome is assessed by at least three assessment tools:

Assessment Tools	Course Outcomes							
	1	2	3	4	5	6	7	8
Programming Projects	•	•		•	•	•	•	
Exams	•	•	•	•	•		•	•
Homework/Quizzes	•	•	•	•	•		•	•
Peer Evaluation						•		
Self-assessment	•	•	•	•	•	•	•	•

*Details of the self-assessment:* Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. At the end of the semester, an anonymous survey of the class is conducted where each student is asked to judge how well each outcome was achieved, using five-point Likert items. The students are also asked to provide recommendations about how to better achieve the Course Outcomes. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

## How Data in the Course is Used to Assess Program Outcomes

Each Course Outcome directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). The data for assessing the Course Outcomes is also used for assessing the corresponding Program Outcomes. For CS 232, Program Outcomes a, b, c, d, e, i, and j are supported.

## Estimated Curriculum Category Content

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0	0	Software design	0.5	0
Data structures	0.5	0	Concepts of programming languages	2.0	0
Computer organization and architecture	0	0		0	0

## Oral and Written Communications

Written reports are part of some laboratory assignments

## Social and Ethical Issues

None.

## COURSE DESCRIPTION

Department and Course Number: CS 260  
Semester Hours: 3.0

Course Title: Data Structures  
Course Coordinator: Jin Soung Yoo

### Current Catalog Description

Design and analysis of data structures, the fundamental algorithms that act upon them, and their implementation as objects. The topics include linear data structures including linked lists, stacks and queues, graphs, trees, heaps, searching, and sorting.

### Textbook

Michael Main, *Data Structures and Other Objects Using Java*, 2<sup>nd</sup> edition, Addison-Wesley (2003).

### References

None.

### Course Outcomes

Upon successful completion of the course requirements, a student should be able to:

1. Recognize the need for data structures and have an ability to choose the appropriate data structure (b, i).
2. Have an ability to evaluate the performance of an algorithm using Big O (b, j).
3. Have an understanding of linked list, stacks and queue data structures, and apply the structures to applications (a, b, c, i).
4. Have an understanding of trees and heap data structures, and apply them to applications (a, b, c, i, j).
5. Have an understanding of various sorting algorithms and the differences of performance, and implement them (a, b, c, k, j).
6. Have an understanding of basic search algorithms (e.g., linear search, binary search) (a, b, j, k).
7. Have an understanding of tree traversals and graph traversals (a, j).

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1		•							•		
2		•								•	
3	•	•	•						•		
4	•	•	•						•	•	
5	•	•	•							•	•
6	•	•								•	•
7	•									•	

### **Program Outcomes**

- l. An ability to apply knowledge of computing and mathematics appropriate to the discipline.
  - a. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
  - b. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
  - c. An ability to function effectively on teams to accomplish a common goal.
  - d. An understanding of professional, ethical, legal, security and social issues and responsibilities.
  - e. An ability to communicate effectively with a range of audiences.
  - f. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
  - g. Recognition of the need for and an ability to engage in continuing professional development.
  - h. An ability to use current techniques, skills, and tools necessary for computing practice.
  - i. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
  - j. An ability to apply design and development principles in the construction of software systems of varying complexity.

### **Prerequisites by Topic**

- A course in discrete mathematics is assumed to have been taken before this course
- Understanding fundamental problem-solving techniques
- Understanding of the fundamentals of object-orientation and abstract data types
- Ability to properly implement classes in Java and to write small programs using Java
- Ability to use multidimensional arrays, exceptions, and I/O techniques

### **Major Topics Covered in the Course (class hours)**

- Software engineering issues and algorithm design (2)
- Performance analysis, big-O, and program complexity (3)
- Review of ADTs, objects, and classes (1)
- ADT invariant for a class (1)
- Collection classes: bag and sequence (2)
- Internal implementation of 1-dim and 2-dim arrays (1)
- Sparse vectors and matrices (1)
- Linked list techniques and examples (3)
- Generic classes and iterators for collections (2)
- Stacks, queues, and applications (4)
- Trees and variations (4)
- Heaps and priority queues (2)
- Searching methods (2)
- Sorting methods (3)
- Directed and undirected graphs, graph traversal, and path algorithms (3)
- Exams (4)

### **Assessment Plan for the Course**

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree",

and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

**How Data in the Course is Used to Assess Program Outcomes**

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 260, Program Outcomes a, b, c, i, j, and k are supported.

**Estimate Curriculum Category Content**

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0.5	0	Software design	0.5	0
Data structures	2.0	0	Concepts of programming languages	0	0
Computer organization and architecture	0	0		0	0

## COURSE DESCRIPTION

Department and Course Number: CS 271  
Semester Hours: 3.0

Course Title: Computer Architecture  
Course Coordinator: Mark C. Temte

### Current Catalog Description

Introduction to computer organization and architecture. Fundamentals of digital logic and representation of numeric and non-numeric data. Assembly-level organization and programming, including instruction formats, addressing modes, and subprogram call/return. Design of main memory, cache memory, and virtual memory. Interrupt basics, interrupt-driven I/O, DMA, and bus protocols. Processor organization, data path, control unit, microprogramming, pipelining, and performance enhancements. Multiprocessor and alternative architectures.

### Textbook

Andrew Tanenbaum, *Structured Computer Organization (5<sup>th</sup> edition)*, Pearson Prentice-Hall, 2006. ISBN 0-13-148521-0.

### References

None.

### Course Outcomes

Upon successful completion of the course requirements, a student should be able to:

1. Demonstrate facility with radix number systems, two's complement arithmetic, and floating-point number representation (a, i).
2. Show familiarity with the Intel 8088 processor and memory model (b, i, j).
3. Show ability to translate basic high-level language control structures into equivalent assembly language statements and to use pseudoinstructions (c, i).
4. Perform assembly language I/O with text, numbers, and files in programs (c, i).
5. Decompose a software problem into a main program and subroutines and implement the solution in assembly language using proper subroutine linkage (c, k).
6. Demonstrate the ability to use the assembly language instruction set and appropriate addressing modes in a program involving a data table (c, i).
7. Show understanding of digital logic concepts and selected combinational logic and sequential logic circuits (a, i).
8. Show understanding of computer memory design, cache concepts, and bus fundamentals (a, b, i, j).
9. Show understanding of the operation of a processor implemented with a microarchitecture (a, j).
10. Demonstrate understanding of processor architecture by writing a portion of an emulator for a processor (b, c, i, k).
11. Show an understanding of interrupts, interrupt handlers, and DMA (b, i).
12. Demonstrate understanding of virtual memory, process concepts, and parallel architectures (b, c, i, j).

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•								•		
2		•							•	•	
3			•						•		
4			•						•		
5			•								•
6			•						•		
7	•								•		
8	•	•							•	•	
9	•									•	
10		•	•						•		•
11		•							•		
12		•	•						•	•	

### Program Outcomes

- An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- An ability to function effectively on teams to accomplish a common goal.
- An understanding of professional, ethical, legal, security and social issues and responsibilities.
- An ability to communicate effectively with a range of audiences.
- An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- Recognition of the need for and an ability to engage in continuing professional development.
- An ability to use current techniques, skills, and tools necessary for computing practice.
- An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- An ability to apply design and development principles in the construction of software systems of varying complexity.

### Prerequisites by Topic

A two-course sequence in high-level language programming is assumed to have been completed before this course together with a course discrete mathematics.

### Major Topics Covered in the Course (class hours)

- Binary arithmetic, two's complement, floating point, and conversion (3 hours)
- Assembly language programming, including registers, memory model, addressing modes, instruction set, conditional instructions, and subprogram linkage (6 hours)
- Two-pass assembly, linking, and debugging (1 hour)
- Input/output devices (3 hours)
- Digital logic: Boolean algebra, gates, combinational and sequential logic (5 hours)
- Memory design (2 hours)
- Bus characteristics, design, and interfacing (3 hours)
- Integration of components into a computer system (1 hour)

9. Design of the CPU and the microarchitecture level (9 hours)
10. Performance enhancements: pipelining and cache memory (3 hours)
11. Traps and interrupts (2 hours)
12. Paging and virtual memory (2 hours)
13. Process concepts and multiprogramming (1 hour)
14. Multiprocessing computers (1 hour)
15. Examinations (4 hours)

**Assessment Plan for the Course**

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

**How Data in the Course is Used to Assess Program Outcomes**

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 271, Program Outcomes a, b, c, i, j, and k are supported.

**Estimate Curriculum Category Content**

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0	0	Software design	0	0
Data structures	0	0	Concepts of programming languages	0.5	0
Computer organization and architecture	2.5	0			

## COURSE DESCRIPTION

Dept. & Course Number : CS 274  
Semester Hours: 3.0

Course Title: Data Communications  
Course Coordinator: David Q. Liu

### Current Catalog Description

This is an introductory level course on the concepts and principles of data communications. Topics include communication media, synchronous and asynchronous transmission, coding, error detection and correction, communication protocol and formats, modulation and demodulation, multiplexing and networking, and the OSI model with emphasis on the physical and data link layers..

### Textbook

Kurose and Ross, *Computer Networking*, Addison Wesley (2008). ISBN 0-321-49770-8.

### Course Outcomes

Upon successful completion of the course requirements, a student should be able to:

1. Understand network architecture, protocols, and design principles (a, e)
2. Use a packet sniffer to observe the data traffic and packet format (i)
3. Write a client-serve application (c, k)
4. understand IP routing mechanism and configure a router (a, c )
5. Understand the multiple access protocols (a, b)

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•				•						
2									•		
3			•								•
4	•		•								
5	•	•									

### Program Outcomes

- a. An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- c. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- d. An ability to function effectively on teams to accomplish a common goal.
- e. An understanding of professional, ethical, legal, security and social issues and responsibilities.
- f. An ability to communicate effectively with a range of audiences.

- g. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- h. Recognition of the need for and an ability to engage in continuing professional development.
- i. An ability to use current techniques, skills, and tools necessary for computing practice.
- j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- k. An ability to apply design and development principles in the construction of software systems of varying complexity.

**Prerequisites by Topic**

- Knowledge of basic compute architecture
- Knowledge of basic data structures and abstract data types
- Knowledge of statement-level control structures and parameter passing methods

**Major Topics Covered in the Course (course hours)**

Computer Networks and Internet (8 hour)  
 Application Layer (8 hours)  
 Transport Layer (12 hours)  
 Network Layer (12 hours)  
 Link Layer and Local Area Networks (8 hours)  
 Tests (2 hours)

**Assessment Plan for the Course**

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

**How Data in the Course is Used to Assess Program Outcomes**

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 274, Program Outcomes a, b, c, e, i, and k are supported.

**Estimate Curriculum Category Content**

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0	0.5	Software design	0	0.5
Data structures	0	0.5	Concepts of programming languages	0	0
Computer organization and architecture	0	1.5			

## COURSE DESCRIPTION

Department and Course Number: CS 306  
Semester Hours: 3.0 Cr

Course Title: Computer In Society  
Course Coordinator: Peter Ng

### Current Catalog Description

Case study analysis of the social impacts of computerization and networking. Topics include computer ethics, crime, privacy, security, reliability, and vulnerability. Other topics include cyberphilia, cyberphobia, censorship, depersonalization, disenfranchisement, automated decision making, artificial intelligence, cognitive science, and ergonomics. Students present projects applying these issues to today's environment.

### Textbook

Sara Baase, *A Gift of Fire (3<sup>rd</sup> edition) - Social, Legal, and Ethical Issues for Computers and Internet*, Pearson Prentice-Hall, 2008. ISBN **10-13-600848-8**.

### References

Paul DePalma, *Computer in Society 08/09 (3<sup>th</sup> edition)*, McGraw-Hill, 2008. ISBN **978-0-07-352848**.

### Course Outcomes

Upon successful completion of the course requirements, a student should be able to:

1. Apply reading materials and class discussions to the real world of works and the social impacts of computerization and networking (f,g,i).
2. Apply each specific topic discussions and related to products and services in my major of study - such as automated decision making (f,g,h,i).
3. Relate homework assignments/case study analysis to real world issues and problems. Real issues such as: computer ethics, crime, privacy, security, reliability, and vulnerability (e,f,g,h,i).
4. Understand the abuse of computer technology and those who have created it. This makes me worry and creates lot of problems for people, such cyberphilia, cyber phobia, censorship depersonalization, and disenfranchisement (e,f,g,h,i).
5. Understand the laws or legality and internet security that govern and regulate the use of computer and internet technology are lacking far behind (e,f,g,h,i).
6. Understand the technological change is a trade-off in which it creates jobs and eliminates jobs. Every advantage has its own disadvantage (f,g,h,i).
7. Can understand the social and economical changes that have advantages and disadvantages of new technologies and are never distributed evenly among people. They're always losers and winners (e,f,g,h,i).
8. Can understand and have ability to communicate effectively with a range of audiences (f,g,h).
9. Can understand each time technological changes and it created the ecological problems, vast, often unpredictable and largely irreversible. These create an impact of local and global of computing on individual's organization and society (e,f,g,h,i).
10. Can understand and recognize the need for an ability to engage in continuing professional development (g,h,i).
11. Can independently do my own research and investigate on topic discussions in class related to my research topic, using technique, skills, and tools necessary for computing practice, and learn great deals from it (e,f,g,h,i).
12. Can relate my professional ethics to the ethical or professional conduct topic which it has been discussed on the Software Engineering Code of Ethics and ACM Code of conducts (e,f,g,h,i).

## Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-i) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•	•				•	•		•		
2		•		•		•	•	•	•		
3		•		•	•	•	•	•	•		
4		•			•	•	•	•	•		
5		•			•	•	•	•	•		
6		•		•		•	•	•	•		
7		•		•	•	•	•	•	•		
8		•		•		•	•	•	•		
9		•		•	•	•	•	•	•		
10		•		•	•		•	•	•		
11		•		•	•	•	•	•	•		
12		•		•	•	•	•	•	•		

## Program Outcomes

- An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- An ability to function effectively on teams to accomplish a common goal.
- An understanding of professional, ethical, legal, security and social issues and responsibilities.
- An ability to communicate effectively with a range of audiences.
- An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- Recognition of the need for and an ability to engage in continuing professional development.
- An ability to use current techniques, skills, and tools necessary for computing practice.
- An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- An ability to apply design and development principles in the construction of software systems of varying complexity.

## Prerequisites by Topic

Junior class standing

## Major Topics Covered in the Course (class hours)

- Five things we need to know about Technological Change (3 hrs)
- Click Fraud, Internet advertisers think they pay only when an interested customer clicks on their ads (1.5 hr)
- The software Wars – The software development is like procurement, and suffers many of the same woes, including excessive complexity and cost overruns (1.5 hr)

4. The Brain Circulation and the New Face of the Silicon Age (1.5 hr)
5. Computer Software Engineers and the Computer Evolution (1.5 hr)
6. Computers, People, and Social Participation (1.5 hr)
7. Societal Institutions: Laws, Politics, Education and the Military (3 hrs)
8. Risk and Avoiding Risk, and International perspectives and Issues (1.5 hr)
9. Unwrapping the Gift – The Ubiquity of Computer and the Rapid Pace of Change (3 hrs)
10. Privacy and Personal Information (3 hrs)
11. Encryptions and Interception of Communication (1.5 hr)
12. Can we Trust the Computer (1.5 hr)
13. Freedom of Speech in Cyberspace (3 hrs)
14. Intellectual Property (3 hrs)
15. Computer Crime (3 hrs)
16. Computer and Work (3 hrs)
17. Broader Issues on the Impact and Control of Computers (2 hrs)
18. Professional Ethics and Responsibility (3 hrs)
19. Project Presentation and Journal Review (1 hrs)
20. Analysis term papers 20 papers (2 hrs)
21. Two Exams (2 hrs)

### Assessment Plan for the Course

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

### How Data in the Course is Used to Assess Program Outcomes

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 306, Program Outcomes a, b, c d, e, f, g, h, and i are supported.

### Estimate Curriculum Category Content

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0	0	Software design	0	0
Data structures	0	0	Concepts of programming languages	0.0	0
Computer organization and architecture	0	0	Analysis of the social, legal, ethical, and educational issues which impact of computerizations and networking.	3.0	

## COURSE DESCRIPTION

Dept. & Course Number : CS 321  
Semester Hours: 3.0

Course Title: Intro to Computer Graphics  
Course Coordinator: Beomjin Kim

### Current Catalog Description

This is an introductory course in Computer Graphics. This course introduces fundamental concepts of computer graphics technology and principles to create three-dimensional graphics. Fundamental graphics algorithms are discussed, as well as graphics programming, using a modern graphics standard. Students are expected to complete several programming assignments that implement fundamental computer graphics techniques in the Window and Linux operating system environments.

### Textbook

- Edward Angel, *Interactive Computer Graphics: A Top-Down Approach Using OpenGL*, 5<sup>th</sup> Ed., Addison-Wesley Publishing, ISBN: 0321535863, 2008.

### References

- Dave Shreiner, Mason Woo, et. al, *OpenGL Programming Guide: The Official Guide to Learning OpenGL*, Version 2.1, 6<sup>th</sup> Ed., Addison-Wesley Publishing, ISBN: 0321481003, 2007.
- Donald Hearn, M. Pauline Baker, *Computer Graphics with OpenGL*, 3<sup>rd</sup> Ed., Prentice Hall, ISBN: 0-13-015390-7, 2003.

### Course Outcomes

The goal of this course is to introduce the student to fundamental concepts of computer graphics technology and principles to create three-dimensional graphics. The letters in parentheses refer to ABET Program Learning Outcomes. A student who successfully fulfills the course requirements will have demonstrated:

1. an understanding of fundamental concepts of computer graphics technology and principles to create three-dimensional graphics (a, h, j)
2. an understanding of basic concepts of computer graphics algorithms (a, j)
3. an understanding of the fundamental mathematics involved in generating a three-dimensional scene (a, j)
4. an ability to use a modern graphics standard for graphics programming (b, i, k)
5. an ability to apply gained graphics programming knowledge to implement graphics applications (b, c, i, k)
6. develop graphics applications in the Window and Linux operating system environments (b, c, i, k)

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•							•		•	
2	•									•	
3	•									•	
4		•							•		•
5		•	•						•		•
6		•	•						•		•

### Program Outcomes

- a. An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- c. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- d. An ability to function effectively on teams to accomplish a common goal.
- e. An understanding of professional, ethical, legal, security and social issues and responsibilities.
- f. An ability to communicate effectively with a range of audiences.
- g. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- h. Recognition of the need for and an ability to engage in continuing professional development.
- i. An ability to use current techniques, skills, and tools necessary for computing practice.
- j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- k. An ability to apply design and development principles in the construction of software systems of varying complexity.

### Prerequisites by Topic

- Knowledge of basic data structures
- Basic understanding of trigonometry and linear algebra (matrices, vector, transformations)
- Knowledge of programming languages of C, C++, or Java
- Familiar with file I/O and arrays
- Ability to implement a medium-size program

### Major Topics Covered in the Course (course hours)

- Introduction to Computer Graphics & C-Language Review (3 hours)
- GLUT Library & Event-driven Programming (1.5 hours)
- Introduction to OpenGL & Graphics Programming (3 hours)
- 3D Object Representations (1.5 hours)
- 2D, 3D, and compound Geometric Transformations (4.5 hours)
- Window-to-Viewport Mapping (1.5 hours)
- Projections (3 hours)
- Introduction to Linux / Mesa 3D graphics library (3 hours)
- Shading (4.5 hours)
- Raster display (3 hours)
- Texture Mapping (3 hours)

- Hidden surface removal (3 hours)
- Clippings (3 hours)
- Display Lists & Advanced features in C.G (3 hours)
- Final Project Presentation & Discussion (3 hours)
- Exams (3 hours)

**Assessment Plan for the Course**

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

**How Data in the Course is Used to Assess Program Outcomes**

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 321, Program Outcomes a, b, c, h, i, j, and k are supported.

**Estimate Curriculum Category Content**

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0.5	1.0	Software design	0.5	
Data structures		0.5	Concepts of programming languages		0.5
Computer organization and architecture					

## COURSE DESCRIPTION

Dept. & Course Number : CS 331

Course Title: Intro to C++ and Object-Oriented  
Programming

Semester Hours: 3.0

Course Coordinator: Beomjin Kim

### Current Catalog Description

An introduction to the C++ language with emphasis on features supporting object-oriented programming. Fundamental data type and operations. Expression evaluation. Selection and iteration constraints. Functions, procedures, and macro. Standard libraries. Classes: declaration and definition; instances; member functions; constructors and destructors; function overloading; inheritance and polymorphism. Stream input and output. Using classes to encapsulate data structure and implementation details.

### Textbook

- P.J. Deitel, H.M. Deitel, *C++ How to Program*, 6<sup>th</sup> Edition, Prentice Hall, ISBN 0-13-615250-3, 2008.

### References

- Bjarne Stroustrup, *The C++ Programming Language*, 3<sup>rd</sup> Ed., Addison-Wesley, ISBN 0201700735, 2000.
- Scott Meyers, *Effective C++: 55 Specific Ways to Improve Your Programs and Designs*, 3<sup>rd</sup> Ed., Addison-Wesley, ISBN 0321334876, 2005.

### Course Outcomes

The goal of this course is to introduce the student to C/C++ programming with emphasis on features supporting object-oriented programming. The letters in parentheses refer to ABET Program Learning Outcomes. A student who successfully fulfills the course requirements will have demonstrated:

1. an ability to use object-oriented programming in C++ for solving problems. (b, c, d, i, k)
2. an understanding of the use of pointers and reference parameters. (i)
3. an understanding of and an ability to apply basic object-oriented concepts including data abstraction, information hiding, and encapsulation with the C++ programming language. (b, c, e, i)
4. an understanding of the object-oriented features in C++: overloading, inheritance, and polymorphism. (b, c, i)
5. an ability to use generic programming in C++. (b, c, i)
6. an understanding of the stream and file Input/Output, and exceptional handling in C++. (i)
7. an ability to apply object-oriented analysis and design methodologies to solve problems with the C++ programming language. (b, c, d, i, k)
8. an understanding of the design patterns (c, i, j, k)

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1		•	•	•					•		•
2									•		
3		•	•		•				•		
4		•	•						•		
5		•	•						•		
6									•		
7		•	•	•					•		•
8			•						•	•	•

### Program Outcomes

- An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- An ability to function effectively on teams to accomplish a common goal.
- An understanding of professional, ethical, legal, security and social issues and responsibilities.
- An ability to communicate effectively with a range of audiences.
- An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- Recognition of the need for and an ability to engage in continuing professional development.
- An ability to use current techniques, skills, and tools necessary for computing practice.
- An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- An ability to apply design and development principles in the construction of software systems of varying complexity.

### Prerequisites by Topic

- Knowledge & experience of a high-level programming language such as Java
- Knowledge of basic data structures
- Ability to implement a medium-size program

### Major Topics Covered in the Course (course hours)

- Introduction to C/C++ Programming, Introduction to the Visual C++ Programming Environment (3 hours)
- Functions, Function Arguments, Scope, Storage class (4.5 hours)
- Pointers, Arrays, and Strings (3 hours)
- Class & Object, Constructor & Destructor, C++ Program Organization (3 hours)
- Composition, Constant member, this, Static, Dynamic Objects (3 hours)
- Object-oriented Analysis & Design (4.5 hours)
- File & Stream Input/Output (1.5 hours)
- Exception Handling (1.5 hours)
- Operator Overloading, Copy Constructor (3 hours)
- Inheritance (3 hours)

- Generic Programming, Templates, Standard Template Library (6 hours)
- Dynamic Binding and Polymorphism (3 hours)
- Design patterns (3 hours)
- Exams (3 hours)

**Assessment Plan for the Course**

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

**How Data in the Course is Used to Assess Program Outcomes**

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 331, Program Outcomes b, c, d, e, i, j, and k are supported.

**Estimate Curriculum Category Content**

Area	Core	Advanced	Area	Core	Advanced
Algorithms			Software design	0.5	0.5
Data structures	0.5		Concepts of programming languages	0.5	1.0
Computer organization and architecture					

## COURSE DESCRIPTION

Dept. & Course Number : CS 350  
Semester Hours: 3.0

Course Title: Programming Language Design  
Course Coordinator: Mark C. Temte

### Current Catalog Description

A survey of language design issues and their implications for translation and run-time support. Examination of modern programming languages and features: abstract data and control structures, procedures, parameter passing mechanisms, block structuring and scope rules, input/output, and storage management. Models of run time behavior. Comparison of imperative and declarative programming languages.

### Textbook

Robert W. Sebesta, *Concepts of Programming Languages (8<sup>th</sup> edition)*, Pearson Education (2008). ISBN 978-0-321-49362-0.

### References

None.

### Course Outcomes

Upon successful completion of the course requirements, a student should be able to:

1. Show knowledge of the evolution of modern programming languages (a).
2. Demonstrate facility with BNF for specifying programming language syntax (a, i).
3. Specify various control structures using operational semantics (a, i).
4. Compute weakest preconditions using the principles of axiomatic semantics (a, i).
5. Show understanding of issues involving variables: data types, binding, strong typing, and scope (a, b, i, j).
6. Show understanding of various subprogram parameter passing methods (a, b, i, j).
7. Show understanding of the issues involved with reference types: dangling pointers, lost heap-dynamic variables, and garbage collection (a, b, i, j).
8. Evaluate design trade-offs associated with various language features (a, c).
9. Design and implement an application in Smalltalk using a number of cooperating classes (b, c, i, k).
10. Design and implement a lexical analyzer and recursive-descent parser for a simple language (b, c, i, k).
11. Implement an application using the functional language Scheme (b, c, i, k).
12. Implement an application using predicates in the logic-programming language Prolog (b, c, i, k).

## Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•										
2	•								•		
3	•								•		
4	•								•		
5	•	•							•	•	
6	•	•							•	•	
7	•	•							•	•	
8	•		•								
9		•	•						•		•
10		•	•						•		•
11		•	•						•		•
12		•	•						•		•

## Program Outcomes

- An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- An ability to function effectively on teams to accomplish a common goal.
- An understanding of professional, ethical, legal, security and social issues and responsibilities.
- An ability to communicate effectively with a range of audiences.
- An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- Recognition of the need for and an ability to engage in continuing professional development.
- An ability to use current techniques, skills, and tools necessary for computing practice.
- An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- An ability to apply design and development principles in the construction of software systems of varying complexity.

## Prerequisites by Topic

Understanding of . . .

- programming techniques in a high-level language such as Java (with experience)
- recursion
- intrinsic data types: integer, float, character, pointer
- basic data structures: bag, sequence, stack, queue, linked list, tree, graph, hash table
- internal implementation of 1-dim and 2-dim arrays
- subprogram linkage using a stack at the assembly language level
- computer organization: von Neumann architecture, processor, memory
- software performance issues (big-O)

### Major Topics Covered in the Course (class hours)

1. Evolution of programming languages (3 hours)
2. Language evaluation criteria and design trade-offs (2 hours)
3. Formal language introduction, BNF, syntax, and semantics (6 hours)
4. Lexical and syntax analysis (3 hours)
5. Names, bindings, type checking, and scopes (3 hours)
6. Data types, pointers, and garbage collection (3 hours)
7. Expressions and assignment statements (3 hours)
8. Statement-level control structures (2 hours)
9. Subprograms, parameter passing, implementation, and generics (4 hours)
10. Smalltalk and object-oriented programming languages (4 hours)
11. Functional programming, LISP, Scheme, and Haskell (6 hours)
12. Logic programming and Prolog (3 hours)
13. Exams (4 hours)

### Assessment Plan for the Course

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

### How Data in the Course is Used to Assess Program Outcomes

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 350, Program Outcomes a, b, c, i, j, and k are supported.

### Estimate Curriculum Category Content

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0	0	Software design	0	0
Data structures	0	0.5	Concepts of programming languages	0.5	2.0
Computer organization and architecture	0	0			

## COURSE DESCRIPTION

Dept., Number	CS 360	Course Title	Software Engineering
Semester hours	3	Course Coordinator	R. L. Sedlmeyer

### Current Catalog Description

P: CS 260 and ENG W234.

An introduction to software engineering using an object-oriented approach. The software development process. Iterative and incremental development. Team organization and project management. Object-oriented analysis and design. Representation of software models using UML: use cases, class and interaction diagrams, and state-charts. Metrics for design evaluation. Software quality assurance. Testing planning and specification; unit and integration test methods. Software tools for analysis and design.

### Textbook

Roger S. Pressman, Software Engineering, A Practitioner's Approach, McGraw Hill, Sixth Edition, ISBN 0-07-285318-2.

### References

1. [Software Engineering: Theory and Practice, 4th Edition](#), by Shari Lawrence Pfleeger and Joanne M. Atlee, 2009.
2. Software Engineering: Principles and Practice, 3rd edition, by [Hans van Vliet](#), Wiley, 2008.
3. Software Engineering, Ian Sommerville, 8th edition, Addison Wesley, 2006.
4. [Schaum's Outline of Software Engineering](#) by David Gustafson, McGraw-Hill, 2002.

### Course Outcomes

1. Gain knowledge of the historical background and fundamental principles of software engineering.(a, h)
2. Gain an understanding of the process of software development and the variety of contemporary development models. (a, h)
3. Learn and apply skills critical to software development with a focus on object-oriented analysis, design, implementation, and testing. (a, b, c, i)
4. Understand the advantages and disadvantages of alternative team organizations for software development. (d)

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•							•			
2	•							•			
3	•	•	•						•	•	•
4				•							

## **Program Outcomes**

- a. An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- c. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- d. An ability to function effectively on teams to accomplish a common goal.
- e. An understanding of professional, ethical, legal, security and social issues and responsibilities.
- f. An ability to communicate effectively with a range of audiences.
- g. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- h. Recognition of the need for and an ability to engage in continuing professional development.
- i. An ability to use current techniques, skills, and tools necessary for computing practice.
- j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- k. An ability to apply design and development principles in the construction of software systems of varying complexity.

## **Prerequisites by Topic**

1. Java
2. Data Structures
3. Technical Writing

## **Major Topics Covered in the Course**

1. Principles of Software Engineering
2. Software Development Life-Cycle
3. Team-oriented Software Development
4. Software Requirements and Specification
5. Software Design
6. Software Construction
7. Software Validation and Verification

## **Assessment Plan for the Course**

The course utilizes two tools for assessment: Student performance on a large group project (which will be completed in CS460), typically solicited from an external client, and the results of a student survey. This course requirement provides an objective measure of level-of-achievement for course outcomes. The project is evaluated by both the instructor and client(s). The survey asks students to judge how well they achieved each learning outcome on a 5-pt Likert scale. The instructor is given the results and encouraged to make recommendations for the next offering, especially for those course outcomes whose average objective or subjective scores fall below 3.

## **How Data in the Course is Used to Assess Program Outcomes (unless adequately covered already in the assessment discussion under Criterion 4)**

The results of the instructor and client project evaluation, together with that of CS460, and student survey are the principal means by which program outcomes are assessed.

**Estimate Curriculum Category Content (Semester hours)**

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0	0	Software design	0	3
Data structures	0	0	Concepts of programming languages	0	0
Computer organization and architecture	0	0			

## COURSE DESCRIPTION (2008 - 2009)

Dept. & Course Number	CS 360	Course Title	Software Engineering
Semester Hours	3.0	Course Coordinator	Peter A. Ng

### Current Catalog Description

This course provides an introduction to the methods of software engineering. Topics include the software development process, software specifications using the UML for analysis and design, software metrics, quality assurance, test methods and plans, and organizational and management issues (including software project management).

### Textbook

Timothy C. Lethbridge and Robert Laganier, *Object-Oriented Software Engineering: Practical Software Development using UML and Java* (2<sup>nd</sup> Edition), McGraw-Hill, ISBN 0-07-70109082.

James A. Whittaker, *How to Break Software – A Practical Guide to Testing*, Addison-Wesley, 2003, ISBN 0-201-79619-8.

### References

- Ian Sommerville, *Software Engineering*, Addison-Wesley (8 Edition) 2006, ISBN 0-201-39815-X .
- Alford, M.W., A Requirements Engineering Methodology for Real Time Processing Requirements, *IEEE Transactions on Software Engineering*, SE-3(1), 1977, p. 60-9.
- Andrew P. Sage and James D. Palmer, *Software Systems Engineering*, John Wiley and Sons.
- Philip Gilbert, *Software Design and Development*, Science Research Associates, Inc.
- Edward Yourdon and Larry Constantine, *Structured Design*, Prentice-Hall.
- Capers Jones, *Software Quality: Analysis and Guidelines for Success*, International Thomson Computer Press, Boston, MA, 1997.
- Ivan Jacobson, Martin Griss and Patrick Johnson, *Software Reuse*, ACM Press, NY, 1997.
- Coad, P. and Yourdon, E., *Object-oriented Analysis*, Englewood Cliffs, NJ, Prentice Hall, 1990.
- Davis, A. M., *Software Requirements: Analysis and Specification*, Englewood Cliffs, NJ, Prentice Hall, 1990.
- DeMarco T., *Structured Analysis and System Specification*, New York, Yourdon Press, 1978.
- Stephen H. Kan, *Metrics and Models in Software Quality Engineering*, Addison-Wesley Publishing Co., Reading MA, 1995.
- Fagan, M.E., Design and Code Inspections to Reduce Errors in Program Development, *IBM Systems Journal*, 15(3) 1976, p.182-211.
- Fagan, M.E., Advances in Software Inspections, *IEEE Transaction on Software Engineering*, SE-12(7), 1986, P.744-751.
- Gilb, T. and Graham, D., *Software Inspection*, Wokingham, Addison-Wesley, 1993.
- Graham, I., *Object-oriented Methods*, 2<sup>nd</sup> eds., Wokingham, Addison-Wesley, 1994.
- Guttag, J., Abstract Data Types and the Development of Data Structures, *CACM*, 26(10) 1977, p.369-405.
- Humphrey, W.S., *Managing the Software Process*, Reading, MA, Addison-Wesley, 1989.
- *IEEE Recommended Practice for Software Requirements Specifications*. In *Software Requirements Engineering* (R. H. Thayer and M. Dorfman, eds), Los Alamitos, CA, IEEE Computer Society Press, 1993.
- *IEEE Software Engineering Standards Collection*, Los Alamitos CA, IEEE Press, 1994.
- Jackson, M.A., *Requirements and Specifications*, Wokingham, Addison Wesley, 1995.
- Liskov, B. and Guttag, J., *Abstraction and Specification in Program Development*, Cambridge, MA, MIT

- Press, 1986.
- Luqi, Computer-aided prototyping for a Command and Control System Using CAPS, IEEE Software, 9(1), 1992, p. 56-67.
  - Mills, H.D., et al, Cleanroom Software Engineering, IEEE Software 4(5) 1987, [19-25.
  - Musa, J.D., Software Reliability Engineering: More Reliable Software, Faster Development and Testing, McGraw-Hill, NY, 1998.
  - Meyers G. J., Reliable Software through Composite Design, Petrocelli, NY, 1975.
  - Neilsen, J., Usability Engineering, Academic Press, NY, 1993.
  - Thayer, R.H. and Dorfman, M., Software Requirements Engineering, 2<sup>nd</sup> edition. Los Alamitos, CA, IEEE Computer Society Press, 1997.
  - Wirth, N., Systematic Programming: An Introduction, Englewood Cliffs, NJ, Prentice Hall, 1976.
  - Boehm, B.W., Software Engineering Economics, Prentice-Hall, Englewood Cliffs, NJ, 1981.
  - Brooks, F.P., The Mythical Man Month, Reading MA, Addison-Wesley, 1975.
  - Booch G., Object-oriented Analysis and Design with Applications, Benjamin Cummings, Redwood City, CA, 1994.

### Course Outcomes

Upon successful completion of the course requirements, a student should be able to:

1. Learn and gain practical experience and theoretical background on the entire software life cycle. [ABET Criterion 3: Program Outcomes: (a), (j-CS), (k-CS)]
2. Learn and utilize the following skills critical to software development: communication, management, feasibility, planning, analysis (software requirements and specifications), design (software design and implementation), quality assurance (software testing, verification and validation), and others (deployment and maintenance). [(b), (c), (i), (j-CS), (k-CS)]
3. Gain an appreciation for the process of software development and the variety of contemporary development models and tools. [(i), (j-CS), (k-CS)]
4. Gain experience for developing deliverable products (fairly large software systems) via team efforts. [(d), (j-CS), (k-CS)]
5. Understand the importance of both process and product in software development. [(g), (h)]
6. Understanding the current practices as well as looking towards future developments. [(i), (h)]
7. Exercise communication skills including oral presentations and written reports to the customers, classmates, and instructor. [(d), (f)]
8. Exercise communication and team participation including team discussion participation, sharing required skills and competency of the subjects, written assigned documents as a team member. [(d), (f), (i)]
9. Be exposed to issues of ethics and professionalism. [(e)]
10. Be able to continue the SW Development process starting from detailed design, quality assurance and others in CS 460. [(h), (j-CS), (k-CS)]

## Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•									•	•
2		•	•						•	•	•
3									•	•	•
4				•						•	•
5							•	•			
6								•	•		
7				•		•					
8				•		•			•		
9					•						
10								•		•	•

### Program Outcomes

- An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- An ability to function effectively on teams to accomplish a common goal.
- An understanding of professional, ethical, legal, security and social issues and responsibilities.
- An ability to communicate effectively with a range of audiences.
- An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- Recognition of the need for and an ability to engage in continuing professional development.
- An ability to use current techniques, skills, and tools necessary for computing practice.
- An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- An ability to apply design and development principles in the construction of software systems of varying complexity.

### Prerequisites by Topic

- Data Structures (CS 260)
- Technical Report Writing (ENG W234)

### Major Topics Covered in the Course

<b>Week</b>	<b>Lecture Topics</b>	<b>Assignments</b>
01-03	Problem Definition and Planning	Statement of problem and proposed solutions Implementation plan <b>Reading Assignments</b> Appendix C: System Description 1. Software and Software Engineering 11. Managing the Software Process 12. Review
04-06	Requirements Analysis	Functional system specifications Data requirements <b>Reading Assignments</b> 4. Developing Requirements 7. Focusing on Users and Their Tasks 8. Modeling Interactions and Behavior
07-09	General Design	System specification (overall system architecture) Testing requirements and plan <b>Reading Assignments</b> 5. Modeling with Classes 9. Architecting and Designing Software 3. Basing Software Development on Reusable Technology
10-11	Detailed Design (Mid Term Due)	Detailed system description <b>Reading Assignments</b> 6. Using Design Pattern 9. Architecting and Designing Software
12-13	Coding and Unit Testing	Operational unit-tested programs with test data (optional) <b>Reading Assignments</b> 10. Testing and Inspecting to Ensure High Quality
14-15	Comprehensive Systems Testing	Program level documentation (optional)
16	System Installation and Turnover System Maintenance	Final Term Project
<b>Week</b> None	<b>Laboratory Outline</b>	

### Assessment Plan for the Course

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

**How Data in the Course is Used to Assess Program Outcomes (unless adequately covered already in the assessment discussion under Criterion 4)**

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 360, Program Outcomes a, through k are supported.

**Estimate Curriculum Category Content (Semester hours)**

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0	0	Software design	0	3.0
Data structures	0	0	Concepts of programming languages	0	0
Computer organization and architecture	0	0			

## COURSE DESCRIPTION

Department and Course Number : CS 364

Course Title: Introduction to  
Database Systems

Semester Hours: 3.0

Course Coordinator: Jin Soung Yoo

### Current Catalog Description

An introductory course on database systems. The topics include entity-relationship model, relational database model, relational algebra, schema normalization, and transaction control, structured query language (SQL), database application development, and additional selective topics such as views, triggers and database procedures.

### Textbook

Raghu Ramakrishnan, Johannes Gehrke, *"Database Management Systems"*, McGraw-Hill Science, ISBN 0072465638, 2002

Jeffrey D. Ullman, Jennifer Widom, *A First Course in Database Systems*, 3<sup>rd</sup> Edition, ISBN 013600637, 2007

### References

None.

### Course Outcomes

Upon successful completion of the course requirements, a student should be able to:

1. Recognize the need for databases. (b, c, i, j).
2. Have an ability to translate real-world requirements into an ER diagram (b, c, i, j, k).
3. Have an ability to translate an ER diagram into relational schemas (a, b, c, i, j).
4. Have an ability to implement real-world queries with SQL queries (a, b, c, i, j)
5. Have an ability to write high-level programming language code embedded with SQL for database applications (b, c, i, j).
6. Have ability to normalize a relational schema (a, b, i, j).
7. Have an understanding of how transaction control in databases works. (a, b, i, j)
8. Have an understanding of other selective topics covered such as view, index, and triggers (a, b, i, j)

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1		•	•						•		
2		•	•						•	•	•
3	•	•	•						•	•	
4	•	•	•						•	•	
5		•	•						•	•	
6	•	•							•	•	
7	•	•							•	•	
8	•	•							•	•	

## **Program Outcomes**

- a. An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- c. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- d. An ability to function effectively on teams to accomplish a common goal.
- e. An understanding of professional, ethical, legal, security and social issues and responsibilities.
- f. An ability to communicate effectively with a range of audiences.
- g. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- h. Recognition of the need for and an ability to engage in continuing professional development.
- i. An ability to use current techniques, skills, and tools necessary for computing practice.
- j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- k. An ability to apply design and development principles in the construction of software systems of varying complexity.

## **Prerequisites by Topic**

- Understanding fundamental problem-solving techniques
- Programming experience in a programming language.
- Knowledge of basic data structures.

## **Major Topics Covered in the Course (class hours)**

- Overview of database systems (2)
- Entity-Relationship Model (4)
- Relational model (4)
- Relational algebra (2)
- Normalization (3)
- SQL: Queries, constraints, triggers (6)
- Database application development (4)
- Transaction management (2)
- Views and index (2)
- Exams (4)

## **Assessment Plan for the Course**

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

### How Data in the Course is Used to Assess Program Outcomes

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 364, Program Outcomes a, b, c, i, j, and k are supported.

### Estimate Curriculum Category Content

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0.5	0.5	Software design	0.5	0
Data structures	0.5	0	Concepts of programming languages		1.0
Computer organization and architecture	0	0		0	0

## COURSE DESCRIPTION

Department and Course Number : CS 365

Course Title: Advanced

Database Systems

Semester Hours: 3.0

Course Coordinator: Jin Soung Yoo

### Current Catalog Description

This course builds upon CS 364 and introduces advance topics in database systems for undergraduates. The topics cover the concepts of database storage, index structures, query optimization, physical database tuning, database architecture, database security, object-relational database, and data warehousing. It also includes introduction to selective emerging database topics or an advanced application development.

### Textbook

Raghu Ramakrishnan, Johannes Gehrke, *"Database Management Systems"*, McGraw-Hill Science, ISBN 0072465638, 2002

### References

David C. Kreines , *"Oracle SQL: the Essential Reference"*, O'Reilly Media, Inc, ISBN 1565926978, 2000

### Course Outcomes

Upon successful completion of the course requirements, a student should be able to:

1. Have an understanding of database storage and various index structures. (a, j)
2. Have an understanding of the basic concepts of database query evaluation and optimizer. (a, j)
3. Have an understanding of physical database tuning for database performance. (a, b, c, j, k)
4. Have an understanding of characteristics of object relational database systems (a, j)
5. Have an ability to apply database security mechanism (a, e)
6. Design a simple database warehouse, and write basic analytical queries(a, b, g, h)
7. Be exposed to emerging database topics such as data mining and information retrieval (a, b, g, h, j)
8. Have an ability to read basic research articles and present the analysis (Optional) (d, f)

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•									•	
2	•									•	
3	•	•	•							•	•
4	•									•	
5	•				•						
6	•	•					•	•	•		
7	•	•					•	•		•	
8				•		•					

## **Program Outcomes**

- a. An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- c. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- d. An ability to function effectively on teams to accomplish a common goal.
- e. An understanding of professional, ethical, legal, security and social issues and responsibilities.
- f. An ability to communicate effectively with a range of audiences.
- g. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- h. Recognition of the need for and an ability to engage in continuing professional development.
- i. An ability to use current techniques, skills, and tools necessary for computing practice.
- j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- k. An ability to apply design and development principles in the construction of software systems of varying complexity.

## **Prerequisites by Topic**

- A course in introduction to database systems is assumed to have been taken before this course
- Understanding relational database systems.
- Knowledge of relational database design.
- Programming experience in SQL and a high-level programming language.
- Knowledge of basic data structures.

## **Major Topics Covered in the Course (class hours)**

- Database architecture (2)
- Database storage (2)
- Index structures and the search method (5)
- Query Evaluation and Optimizer (5)
- Physical Database Tuning (4)
- Security and Authorization (2)
- Object relational database systems (4)
- Introduction to data warehousing (4)
- Introduction to emerging database topics (5)
- Exams (4)

## **Assessment Plan for the Course**

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

### How Data in the Course is Used to Assess Program Outcomes

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 365, Program Outcomes a – k are supported.

### Estimate Curriculum Category Content

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0.5	0.5	Software design	0.5	0
Data structures	0	0.5	Concepts of programming languages	0	0.5
Computer organization and architecture	0	0.5		0	0

## COURSE DESCRIPTION

Dept., Number	CS 368	Course Title	Human Computer Interaction
Semester hours	3	Course Coordinator	R. L. Sedlmeyer

### Current Catalog Description

Introduction to general issues surrounding human computer interaction (HCI). The course presents principles, design methodologies, tools, and evaluation techniques with an emphasis on human-centered interface design and development. Other issues covered include HCI aspects of multimedia systems, World Wide Web, computer-supported cooperative work, and recent paradigms of HCI.

### Textbook

User Interface Design and Evaluation, Debbie Stone et al, Morgan Kaufmann, 2005, ISBN: 0-12-088436-4.

### References

1. Eclipse Rich Client Platform: Designing, Coding and Packaging Java Applications, J. McAffer and J-M Lemieux, Addison Wesley, 2006, ISBN: 0-321-33461-2.
2. The Elements of User Interface Design, Theo Mandel, John Wiley & Sons, 1997.
3. GUI Design Essentials, Susan Weinschenk, Pamela Jamar, Sarah C. Yeo, John Wiley & Sons, 1997.
4. The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques, Wilbert O. Galitz, John Wiley & Sons, 2007.
5. About Face 3.0: The Essentials of User Interface Design, Alan Cooper and Robert Reimann, John Wiley and Sons, 2007.
6. Human-Computer Interaction, Alan Dix, Janet Finlay, Gregory Abowd, Russel Beale, Prentice Hall, 2004.
7. Usability Inspection Methods, Jakob Nielsen, Robert L. Mack (Editors), John Wiley & Sons, 1994.

### Course Outcomes

1. Understanding of basic concepts in human-computer interaction and the characteristics of a graphical user interface (a, g)
2. Awareness of the cognitive psychology issues associated with human-computer interaction (a, f)
3. Application of GUI design principles and standards (c)
4. Analysis of user interface requirements (a, b, d, f, g, i)
5. Design of graphical user interfaces with a focus on WIMP interfaces (a, c, d)
6. Implementation of graphical user interfaces using an API. (c, d, i, k)
7. Understanding of interface evaluation issues and usability testing techniques. (c, d, f, g, i)

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•						•				
2	•					•					
3			•								
4	•	•		•		•	•		•		
5	•		•	•							
6			•	•					•		•
7			•	•		•	•		•		

### Program Outcomes

- An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- An ability to function effectively on teams to accomplish a common goal.
- An understanding of professional, ethical, legal, security and social issues and responsibilities.
- An ability to communicate effectively with a range of audiences.
- An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- Recognition of the need for and an ability to engage in continuing professional development.
- An ability to use current techniques, skills, and tools necessary for computing practice.
- An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- An ability to apply design and development principles in the construction of software systems of varying complexity.

### Prerequisites by Topic

- Java
- Data Structures

### Major Topics Covered in the Course

- Foundations of human-computer interaction
- Human-centered software evaluation
- Human-centered software development
- Graphical user-interface design
- Graphical user-interface programming

### Assessment Plan for the Course

The course utilizes two tools for assessment: Student performance on a group project that incorporates all aspects of the human-centered SDLC covered in the course and the results of a student survey. This course requirement provides an objective measure of level-of-achievement for course outcomes. The survey asks students to judge how well they achieved each learning outcome on a 5-pt Likert scale. The instructor is given the results and encouraged

to make recommendations for the next offering, especially for those course outcomes whose average objective or subjective scores fall below 3.

**How Data in the Course is Used to Assess Program Outcomes (unless adequately covered already in the assessment discussion under Criterion 4)**

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 368, Program Outcomes a, b, c, d, f, g, i, and k are supported.

**Estimate Curriculum Category Content (Semester hours)**

Area	Core	Advanced	Area	Core	Advanced
Algorithms			Software design		2.5
Data structures			Concepts of programming languages		0.5
Computer organization and architecture					

## COURSE DESCRIPTION

Dept., Number	CS 372	Course Title	Web Application Development
Semester hours	3	Course Coordinator	R. L. Sedlmeyer

### Current Catalog Description

P: CS 274. Introduction to Web application development. Characteristics of Web and application servers; Web engineering principles and application architectures; Web page construction; client and server-side scripting; database interaction; Web application deployment and management; security and performance issues; overview of application-layer protocols.

### Textbook

Web Technologies: A Computer Science Perspective, by Jeffrey C. Jackson, Pearson Prentice Hall, 2007, ISBN 0-13-185603-0

### References

1. Core servlets and JavaServer Pages, Marty Hall, Larry Brown, Prentice Hall PTR, 2003.
2. HTML, XHTML and CSS, Rob Huddlestone, Wiley Publishing, Inc., 2008.
3. [www.w3schools.com](http://www.w3schools.com) tutorials on HTML and CSS

### Course Outcomes

1. Describe the essential concepts associated with internet architecture that enable web applications (i)
2. Describe the basic components of web browsers and servers (c)
3. Construct a web server that implements a small subset of HTTP (a, c, j)
4. Utilize an integrated development environment to construct and deploy a web application (i)
5. Construct and validate web pages using HTML 4.01, XHTML 1.0 and CSS 2.1 (a, c, i)
6. Implement client-side application logic using Javascript (a, b, c)
7. Implement server-side application logic using Java servlets (a, b, c)
8. Implement the model-view-controller architecture using Java Server Pages (a, c, j, k)
9. Utilize JDBC to interact with persistent storage (a, c)
10. Read and apply web standards documents (a, h, i)
11. Identify trends in web technologies (h)
12. Independently investigate and apply non-Java technologies to construct web applications (a, b, c, h, i, k)

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1									•		
2			•								
3	•		•							•	
4									•		
5	•		•						•		
6	•	•	•								
7	•	•	•								
8	•		•							•	•
9	•		•								
10	•							•	•		
11								•			
12	•	•	•					•	•		•

### Program Outcomes

- An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- An ability to function effectively on teams to accomplish a common goal.
- An understanding of professional, ethical, legal, security and social issues and responsibilities.
- An ability to communicate effectively with a range of audiences.
- An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- Recognition of the need for and an ability to engage in continuing professional development.
- An ability to use current techniques, skills, and tools necessary for computing practice.
- An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- An ability to apply design and development principles in the construction of software systems of varying complexity.

### Prerequisites by Topic

- Java
- Socket programming
- HTTP protocol

### Major Topics Covered in the Course

- HTML and XHTML.
- CSS
- Java Servlets
- Java Server Pages
- Javascript
- Web Browser and Server Functions and Architecture

### Assessment Plan for the Course

The course utilizes two tools for assessment: Student performance on a mid-sized project that incorporates all technologies covered in the course and the results of a student survey. This course requirement provides an objective measure of level-of-achievement for course outcomes. The survey asks students to judge how well they achieved each learning outcome on a 5-pt Likert scale. The instructor is given the results and encouraged to make recommendations for the next offering, especially for those course outcomes whose average objective or subjective scores fall below 3.

### How Data in the Course is Used to Assess Program Outcomes (unless adequately covered already in the assessment discussion under Criterion 4)

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 372, Program Outcomes a, b, c, h, i, j, and k are supported.

### Estimate Curriculum Category Content (Semester hours)

Area	Core	Advanced	Area	Core	Advanced
Algorithms			Software design		2.5
Data structures		0.25	Concepts of programming languages		0.25
Computer organization and architecture					

## COURSE DESCRIPTION

Dept. & Course Number: CS 374  
Semester Hours: 3.0

Course Title: Computer Networks  
Course Coordinator: David Q. Liu

### Current Catalog Description

The design and implementation of data communications networks. Topics include network topologies; messages, circuit and packet switching; broadcast, satellite and local area networks; routing; the OSI model with emphasis on the network, transport, and session layers.

### Textbook

Peterson and Davie, *Computer Networks A Systems Approach, 4th edition*, Morgan Kaufmann (2008). ISBN 0-12-574013-4.

### Course Outcomes

Upon successful completion of the course requirements, a student should be able to:

1. Understand network architecture, protocols, and design principles (a, b, e)
2. Understand and apply the principle of internetworking (a, b, c, i)
3. Understand and apply the end-to-end principle (a, c, i)
4. Apply resource allocation techniques for network designs (a, c, i, j, k)
5. Understand mobile and satellite communication systems (a, b)
6. Understand and utilize multimedia applications (a, b, g)

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•	•			•						
2	•	•	•						•		
3	•		•						•		
4	•		•						•	•	•
5	•	•									
6	•	•					•				

### Program Outcomes

- a. An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- c. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- d. An ability to function effectively on teams to accomplish a common goal.
- e. An understanding of professional, ethical, legal, security and social issues and responsibilities.
- f. An ability to communicate effectively with a range of audiences.
- g. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- h. Recognition of the need for and an ability to engage in continuing professional development.

- i. An ability to use current techniques, skills, and tools necessary for computing practice.
- j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- k. An ability to apply design and development principles in the construction of software systems of varying complexity.

**Prerequisites by Topic**

- Knowledge of basic compute architecture
- Knowledge of basic operating systems
- Knowledge of basic data communication systems
- Knowledge of basic algorithm analysis

**Major Topics Covered in the Course (course hours)**

Foundation for computer networks and Directed link networks (9 hours)  
 Packet Switching (3 hours)  
 Internetworking (6 hours)  
 End-to-end protocols (3 hours)  
 Congestion Control (3 hours)  
 Resource Allocation (6 hours)  
 Multimedia Applications (6 hours)  
 Mobile Communication Systems (6 hours)  
 Satellite Systems (3 hours)  
 Tests (2 hours)

**Assessment Plan for the Course**

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

**How Data in the Course is Used to Assess Program Outcomes**

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 374, Program Outcomes a, b, c, e, i, and k are supported.

**Estimate Curriculum Category Content**

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0	1.0	Software design	0	0.5
Data structures	0	0.5	Concepts of programming languages	0	0
Computer organization and architecture	0	1.0			

## COURSE DESCRIPTION

Department and Course Number : CS 380  
Semester Hours: 3.0

Course Title: Artificial Intelligence  
Course Coordinator: Jin Soung Yoo

### Current Catalog Description

Introduction of the basic principles in artificial intelligence area. The topics include agents, search strategies, constraint satisfaction problems, game playing, logic, inference, knowledge representation, planning, a traditional AI language, and some of selective advanced topics.

### Textbook

Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, Second Edition, Prentice Hall, ISBN 0-13-790395-2, 2003.

### References

Ben Coppin , *Artificial Intelligence Illuminated*, Jones and Bartlett Publishers, ISBN 0-7637-3230-3, 2004

### Course Outcomes

Upon successful completion of the course requirements, a student should be able to:

1. Have an appreciation for AI and agents. (a).
2. Have an understanding of uninformed search for problem solving. (a, b, j, k).
3. Have an understanding of heuristic search strategies for problem solving. (a, b, j, k).
4. Have an understanding of constraint satisfaction problems (a, b, j, k).
5. Have an understanding of game playing as search problems (a, b, j, k).
6. Have an understanding of logics for knowledge representation, and inference rules. (a, b, j)
7. Have an appreciation of planning problem (a, b, k).
8. Have a basic proficiency in a traditional AI language (a, c, i).

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•										
2	•	•								•	•
3	•	•								•	•
4	•	•								•	•
5	•	•								•	•
6	•	•								•	
7	•	•									•
8	•		•						•		

### Program Outcomes

- a. An ability to apply knowledge of computing and mathematics appropriate to the discipline.

- b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- c. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- d. An ability to function effectively on teams to accomplish a common goal.
- e. An understanding of professional, ethical, legal, security and social issues and responsibilities.
- f. An ability to communicate effectively with a range of audiences.
- g. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- h. Recognition of the need for and an ability to engage in continuing professional development.
- i. An ability to use current techniques, skills, and tools necessary for computing practice.
- j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- k. An ability to apply design and development principles in the construction of software systems of varying complexity.

### **Prerequisites by Topic**

- Understanding fundamental problem-solving techniques
- Knowledge of basic discrete mathematics
- Programming experience in a programming language.
- Knowledge of basic data structures.

### **Major Topics Covered in the Course (class hours)**

- AI and Agents (4)
- Uninformed search (4)
- Heuristic search (4)
- AI programming language (4)
- Constraint Satisfaction Problems (4)
- Game Playing (3)
- Propositional Logic (3)
- First-Order Logic and Resolution (4)
- Knowledge representation (2)
- Planning (2)
- Exams (4)

### **Assessment Plan for the Course**

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

### **How Data in the Course is Used to Assess Program Outcomes**

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 380, Program Outcomes a, b, c, i, j, and k are supported.

**Estimate Curriculum Category Content**

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0.5	0.5	Software design	0.5	0.5
Data structures	0	0	Concepts of programming languages	0.5	0.5
Computer organization and architecture	0	0		0	0

## COURSE DESCRIPTION

Department and Course Number                      CS 384                      Course Coordinator    Gyorgy Petruska

Course Title    Numerical Analysis                      Total Credits    3.0

### **Current Catalog Description**

Iterative methods for solving nonlinear equations; direct and iterative methods solving linear systems; interpolation and extrapolation; approximation of derivatives, integrals, and functions; numerical techniques for ordinary differential equations; error analysis. Use of mathematical subroutine libraries.

### **Textbook**

Numerical Analysis, Richard L. Burden, J. Douglas Faires, 8<sup>th</sup> edition, Brooks/Cole (Required)

### **References**

None

### **Course Goals**

This course introduces the students to the algorithms most frequently applied for the solutions of numerical problems. Starting out with the solution of non-linear equations the course leads to ordinary differential equations and iterative solutions of linear systems each area with the relevant error analysis and error estimation. The students learn the natural limitations of the modern computerized approximation techniques by understanding the effect of truncation and round-off errors in iterations. For implementing the algorithms the students can chose their preferred programming language, iteration tools or Computer Algebra System.

### **Course Learning Outcomes**

The learning outcomes (course objectives) specific for this course are as follows:

1. The ability to construct and implement algorithms to find the root of functions, applying the bisection, Newton, secant and false position methods ((a), (b),(c), (i), (j)); practice and testing in homeworks 1- 2, Exam 1.
2. Applying estimations on various speeds of convergence and to apply acceleration principles ((a), (b), (c), (i), (j)); practice and testing in homework 2, Exam 1.
3. Constructing interpolation polynomials in Lagrange, Newton, Hermit and spline forms ((a), (b),(c), (i), (j)); practice and testing in homeworks 3 - 4, Exam 1.
4. Applying the basic numerical integration and differentiation methods, and constructing Richardson and Romberg matrices ((a), (b),(c), (i), (j)); practice and testing in homeworks 5-6, Exam 1 -2.
5. The ability to solve initial value problems using Runge-Kutta methods of varying orders, and to apply multi-step and predictor- corrector methods ((a), (b),(c), (i), (j)); practice and testing in homeworks 7, Exam 2.
6. Finding solutions of linear systems of equations using Gaussian elimination with scaled partial pivoting and recognizing the fundamental algebraic properties of vectors and matrices as related to system of equations. ((a), (b),(c), (i), (j)); practice and testing in Exam 2.
7. The ability to carry out and utilize error analysis of the approximations ((a), (b),(c),(e), (i), (j)); practice and testing in homework 1- 7, Exam 1 - 2.

### **Relationship between Course Outcomes and Program Outcomes**

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•	•	•						•	•	
2	•	•	•						•	•	
3	•	•	•						•	•	
4	•	•	•						•	•	
5	•	•	•						•	•	
6	•	•	•						•	•	
7	•	•	•		•				•	•	

**Program Outcomes**

- a. An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- c. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- d. An ability to function effectively on teams to accomplish a common goal.
- e. An understanding of professional, ethical, legal, security and social issues and responsibilities.
- f. An ability to communicate effectively with a range of audiences.
- g. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- h. Recognition of the need for and an ability to engage in continuing professional development.
- i. An ability to use current techniques, skills, and tools necessary for computing practice.
- j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- k. An ability to apply design and development principles in the construction of software systems of varying complexity.

**Prerequisites by Topic**

Two semesters of calculus  
 CS 160 or equivalent knowledge of programming.

**Major Topics Covered in the Course (Academic hours)**

- Computer arithmetic, roundoff errors, convergence, mathematical background..... (3)
- Solutions of equations;
  - bisection and fixed point iteration, Newton method, acceleration of convergence... (6)
- Approximation by interpolation;
  - Lagrange, Newton, Hermit polynomials; cubic splines..... (6)
- Numerical differentiation; three and five points rules, Richardson extrapolation..... (4)
- Numerical integration;
  - composite rules, Romberg integration, Gaussian Quadrature..... (5)
- Numerical solutions of ordinary differential equations and systems;
  - Taylor and Runge-Kutta methods, multistep and variable stepsize methods, higher order equations and systems, stiffness..... (9)
- Direct methods for linear systems; Gauss elimination, pivoting methods, matrix manipulation, ill conditioned systems, norms, error bounds..... (6)
- Iterative solution of linear systems; Jacobi, Gauss-Seidel and relaxation methods..... (3)

Exams..... (3)

**Projects**

Homework assignments are required problem solutions of selected book exercises.

**Assessment Plan for the Course**

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

**Credit Content.**

The course requires 3 credit hours for lectures per week (1 credit or academic hour is 50 minutes), making 45 hour per semester. The exams are take-home and require extra time.

**Estimated CSAB Category Content**

A 3 credit-hour class has 45 contact hours.

	COR E	ADVANC ED		COR E	ADVANC ED
Data Structures	0	0	Computer Organization and Architecture	0	0
Algorithms	3	0	Concepts of Programming Languages	0	0
Software Design	0	0			

**Oral and Written Communications**

Written reports are part of the homework assignments

**Oral presentations** not required. Students may volunteer to present their algorithms to the class. The students are commonly required to address questions from the instructor and classmates.

**Social and Ethical Issues**

Professional responsibility is discussed in relation with error analysis and reliability of the computed solutions.

**Theoretical Content**

- Elements of Mathematical Analysis
- Elements of Computer Arithmetic
- Convergence of iterations of operators
- Algebra of Polynomials
- Differential equations, existence and uniqueness of solutions

Linear and matrix algebra

## COURSE DESCRIPTION

Dept. & Course Number : CS 421  
Semester Hours: 3.0

Course Title: Advanced Computer Graphics  
Course Coordinator: Beomjin Kim

### Current Catalog Description

Advanced topics in computer graphics such as three-dimensional rendering, curve and surface design, anti-aliasing, animation, and visualization. Other topics will be selected depending on current research trends. Through development of projects, students will gain practical experience about modern computer graphics.

### Textbook

- Jeffrey J. McConnell, *Computer Graphics: Theory Into Practice*, Jones & Bartlett, ISBN: 0763722502, 2005.

### References

- Alan Watt, *3D Computer Graphics*, 3<sup>rd</sup> Edition, Addison-Wesley Publishing, ISBN: 0201398559, 1999.
- James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, and Richard L. Phillips, *Computer Graphics: Principles and Practice*, Second edition in C, Addison-Wesley Publishing, ISBN: 0-201-84840-6, 1996.

### Course Outcomes

The goal of this course is to introduce the student to advanced concepts of computer graphics technology and principles. The letters in parentheses refer to ABET Program Learning Outcomes. A student who successfully fulfills the course requirements will have demonstrated:

7. an understanding of advanced concepts and principles of computer graphics algorithms (a, i)
8. an ability to use raster image to implement graphics applications. (i)
9. an understanding of the fundamental mathematics involved in curve and curved surface generation (a, i)
10. an ability to implement curve and curved surface (a, i)
11. an understanding of the advanced rendering techniques (a, b)
12. an understanding of the computer animation techniques (i)
13. an understanding of the particle systems (a, i)
14. an ability to apply gained graphics programming knowledge to implement graphics applications (a, b, i, j)

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•								•		
2									•		
3	•								•		
4	•								•		
5	•	•									
6									•		
7	•								•		
8	•	•							•	•	

### Program Outcomes

- An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- An ability to function effectively on teams to accomplish a common goal.
- An understanding of professional, ethical, legal, security and social issues and responsibilities.
- An ability to communicate effectively with a range of audiences.
- An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- Recognition of the need for and an ability to engage in continuing professional development.
- An ability to use current techniques, skills, and tools necessary for computing practice.
- An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- An ability to apply design and development principles in the construction of software systems of varying complexity.

### Prerequisites by Topic

- Knowledge of introductory computer graphics
- Knowledge & experience of graphics algorithms and programming using OpenGL or similar modern graphics library
- Knowledge of basic data structures
- Ability to implement a medium-size program
- Basic understanding of trigonometry and linear algebra (matrices, vector, transformations)

### Major Topics Covered in the Course (course hours)

- Introduction, Raster display (3 hours)
- Line drawing algorithms (3 hours)
- Spline curve design (3 hours)
- Curved surface design (3 hours)
- Particle systems (3 hours)
- Radiosity (6 hours)
- Ray tracing (6 hours)
- Animation (3 hours)
- Research topics in Computer Graphics (9 hours)
- Project Presentation & Discussion (6 hours)
- Exams (2 hours)

### **Assessment Plan for the Course**

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

### How Data in the Course is Used to Assess Program Outcomes

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 421, Program Outcomes a, b, i, and j are supported.

### Estimate Curriculum Category Content

Area	Core	Advanced	Area	Core	Advanced
Algorithms	1.0	1.0	Software design		
Data structures		0.5	Concepts of programming languages		0.5
Computer organization and architecture					

## COURSE DESCRIPTION

Dept. & Course Number : CS 445  
Semester Hours: 3.0

Course Title: Computer Security  
Course Coordinator: David Q. Liu

### Current Catalog Description

A survey of the fundamentals of computer security. Topics include risks and vulnerabilities, policy formation, controls and protection methods, survey of malicious logic, database security, encryption, authentication, intrusion detection, network and system security issues, personnel and physical security issues, security design principles, issues of law and privacy.

### Textbook

Stallings, *Computer Security*, Prentice Hall (2008). ISBN 0-13-600424-0.

### Course Outcomes

Upon successful completion of the course requirements, a student should be able to:

1. develop and implement any symmetric encryption algorithm (a, c, i, k)
2. analyze and identify the possible security issue of a computer system (a, b, j)
3. apply basic skills to develop secure software (a, g, i, j)
4. analyze the legal and ethical issues of computer security (e, f)
5. use public key encryption algorithms for system security (a, i)
6. use PGP (pretty good privacy) based secure email systems (i)
7. apply and use access control mechanisms (i)

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•		•						•		•
2	•	•								•	
3	•						•		•	•	
4					•	•					
5	•								•		
6									•		
7									•		

### Program Outcomes

- a. An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- c. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.

- d. An ability to function effectively on teams to accomplish a common goal.
- e. An understanding of professional, ethical, legal, security and social issues and responsibilities.
- f. An ability to communicate effectively with a range of audiences.
- g. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- h. Recognition of the need for and an ability to engage in continuing professional development.
- i. An ability to use current techniques, skills, and tools necessary for computing practice.
- j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- k. An ability to apply design and development principles in the construction of software systems of varying complexity.

### **Prerequisites by Topic**

- Knowledge of basic compute architecture
- Knowledge of basic data structures and abstract data types
- Knowledge of basic algorithm analysis
- Knowledge of basic operating systems

### **Major Topics Covered in the Course (course hours)**

1. Overview of Computer Security (3 hours)
2. Cryptographic Tools (3 hours)
3. User Authentication (3 hours)
4. Access Control (4 hours)
5. Intrusion Detection (6 hours)
6. Malicious Software (4 hours)
7. Denial of Services (4 hours)
8. Firewalls and Intrusion Prevention Systems (3 hours)
9. Buffer Overflow (3 hours)
10. Legal and Ethical Aspects (3 hours)
11. Symmetric Encryption and Message Confidentiality (4 hours)
12. Public-Key Cryptography and Message Authentication (4 hours)
13. Tests (2 hours)

### **Assessment Plan for the Course**

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

### **How Data in the Course is Used to Assess Program Outcomes**

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 445, Program Outcomes a, b, c, e, f, g, i, j, and k are supported.

**Estimate Curriculum Category Content**

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0	1.0	Software design	0	0.5
Data structures	0	0.5	Concepts of programming languages	0	0.5
Computer organization and architecture	0	0.5			

## **COURSE DESCRIPTION** (2008 - 2009)

Dept. & Course Number	CS 460	Course Title	Capstone Design and Professional Practice
Semester Hours	4.0	Course Coordinator	Urcun (John) Tanik

### **Current Catalog Description**

Student teams will participate in the design and implementation of a substantial software project. Topics include practical issues of software development, quality assurance and deployment, as well as computing ethics and professional practice.

### **Textbook**

- Ian Sommerville, Software Engineering, Addison-Wesley (8<sup>th</sup> Edition) 2007, ISBN 0-321-31379-8
- Timothy C. Lethbridge and Robert Laganriere, Object-Oriented Software Engineering: Practical Software Development using UML and Java (2<sup>nd</sup> Edition), McGraw-Hill, 2005, ISBN 0-07-70109082.

### **References**

- Roger S. Pressman and David Lowe, Web Engineering: A Practitioner's Approach, McGraw-Hill, 2009, ISBN 978-0-07-352329-3
- Christopher Fox, Introduction to Software Engineering Design: Processes, Principles, and Patterns with UML2, Addison-Wesley Computing, 2006. ISBN 0-321-41013-0
- Stephen H. Kan, Metrics and Models in Software Quality Engineering, Second Edition, Addison-Wesley Publishing Co., Reading MA, 2002.
- Guide to Software Engineering Body of Knowledge, 2004, in Sommerville preface, page ix, <http://www.swebok.org>

## Course Outcomes

CS 460 is a continuation of CS360 and covers a variety of topics related to the practice of software engineering. The goals of this course are: (1) To increase student software development skills by completing a substantial software project for a real client working as part of a team; (2) To increase student technical presentation skills; (3) To increase student independent learning skills; and (4) To gain awareness of ethical issues and expectations for professional conduct.

Upon successful completion of the course requirements, a student should be able to:

1. Apply UML to construct various design models, e.g. class, state and interaction models
2. Apply a subset of product metrics to a software model
3. Apply a subset of process and project metrics to support project management
4. Construct a software project schedule and track its progress
5. Perform simple estimates of software cost and level of effort for an object-oriented project
6. Define common software quality assurance activities
7. Conduct a formal technical review
8. Define common elements of a software configuration management (SCM) system
9. Utilize an SCM repository for project artifacts
10. Apply software engineering principles and skills to complete a team-oriented software project
11. Independently investigate a software engineering topic and prepare a professional presentation
12. Independently investigate an ethical or professional conduct topic and prepare a professional presentation
13. Apply industry standards, methods, terminology, and techniques as defined in the international Guide to the Software Engineering Body of Knowledge.
14. Prepare industrial documentation, including an official project management plan (PMP) according to IEEE standard 1058, and deliver professional presentation of team project upon completion.

## Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•								•	•	•
2	•								•		•
3	•								•		•
4	•								•		•
5	•								•		•
6	•								•		•
7						•					•
8	•								•		•
9	•								•		•
10	•	•	•	•					•		•
11	•					•					
12					•	•	•	•			
13	•	•	•	•	•	•		•		•	
14		•		•		•	•				

## Program Outcomes

- a. An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- c. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- d. An ability to function effectively on teams to accomplish a common goal.
- e. An understanding of professional, ethical, legal, security and social issues and responsibilities.
- f. An ability to communicate effectively with a range of audiences.
- g. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- h. Recognition of the need for and an ability to engage in continuing professional development.
- i. An ability to use current techniques, skills, and tools necessary for computing practice.
- j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- k. An ability to apply design and development principles in the construction of software systems of varying complexity.

## Prerequisites by Topic

- Software Engineering (CS 360)
- Senior Standing

## Major Topics Covered in the Course

- During capstone course transition, addressing evolving sponsor needs with agile software development approach to adapt to software lifecycle considerations that encompass architectural design and distributed systems modifications (1.5 hours)
- During capstone course transition, addressing evolving sponsor needs with agile software development approach to adapt to object-oriented design modifications (1.5 hours)
- During capstone course transition, addressing evolving sponsor needs with agile software development approach to adapt to real-time software design modifications (1.5 hours)
- During capstone course transition, addressing evolving sponsor needs with agile software development approach to adapt to user-interface design modifications (1.5 hours)
- During capstone course transition, addressing evolving sponsor needs with agile software development approach to adapt to modifications in industry standards, tools, and methods before deployment (1.5 hours)
- During capstone course transition, addressing evolving sponsor needs with agile software development approach to adapt to validation, verification, metrics, quality assurance, test methods, and plans (1.5 hours)
- Rapid software development, reuse, component-based software engineering (3 hours)
- Critical systems development, security engineering (3 hours)
- Management, software cost estimation and metrics, quality assurance, process improvement (3 hours)
- Software evolution and emerging technologies (3 hours)
- Service-oriented software engineering, aspect-oriented software development (3 hours)
- Deployment, meeting industry standards and ethics, and sponsor satisfaction (4.5 hours)
- Establishing maintenance mechanisms and configuration techniques for sponsor (3 hours)
- Quizzes, presentations, progress reports, and examinations (5 hours)
- Assignments and team meetings (5 hours)
- Final presentation and documentation of consolidated work for sponsor (4.5 hours)
- Laboratory development and sponsor meetings under supervision (15 hours)

**Assessment Plan for the Course**

- Review first week student self-evaluation of course objectives and background
- Review and give feedback on engineering log reporting on progress for project team
- Verify engineering log by reviewing continuously updated concept map displaying online actual work output, in addition to comprehensive relationship between attached documentation
- Track Attendance
- Grade Midterm exam
- Grade Final exam
- Grade official project reviews – proposal, requirements, design, including presentations Periodic evaluation and feedback of progress reports as updated on comprehensive project Concept Map displayed online with most recent versions of attached assignments and their interrelationships for ease of reference and document management
- Grade final project presentation and documentation, including all work in CS 360
- Review course outcomes: Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

**How Data in the Course is Used to Assess Program Outcomes (unless adequately covered already in the assessment discussion under Criterion 4)**

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 460, Program Outcomes a, through k are supported.

**Estimate Curriculum Category Content (Semester hours)**

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0	0	Software design and development	0	4.0
Data structures	0	0	Concepts of programming languages	0	0
Computer organization and architecture	0	0			

## COURSE DESCRIPTION

Dept., Number	CS 460	Course Title	Capstone Design and Professional Practice
Semester hours	4	Course Coordinator	R. L. Sedlmeyer

### Current Catalog Description

P: CS360 and Senior Standing. Student teams will participate in the design and implementation of a substantial software project. Topics include practical issues of software development, quality assurance, and deployment, as well as computing ethics and professional practice.

### Textbook

Roger S. Pressman, Software Engineering, A Practitioner's Approach, McGraw Hill, Sixth Edition, ISBN 0-07-285318-2.

### References

1. Eclipse Rich Client Platform: Designing, Coding and Packaging Java Applications, J. McAffer and J-M Lemieux, Addison Wesley, 2006, ISBN: 0-321-33461-2.
2. The Elements of User Interface Design, Theo Mandel, John Wiley & Sons, 1997.
3. GUI Design Essentials, Susan Weinschenk, Pamela Jamar, Sarah C. Yeo, John Wiley & Sons, 1997.
4. The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques, Wilbert O. Galitz, John Wiley & Sons, 2007.
5. About Face 3.0: The Essentials of User Interface Design, Alan Cooper and Robert Reimann, John Wiley and Sons, 2007.
6. Human-Computer Interaction, Alan Dix, Janet Finlay, Gregory Abowd, Russel Beale, Prentice Hall, 2004.
7. Usability Inspection Methods, Jakob Nielsen, Robert L. Mack (Editors), John Wiley & Sons, 1994.

### Course Outcomes

1. Apply UML to construct class, state and interaction models (a, i, k)
2. Apply a subset of product metrics to a software model (a, i, k)
3. Apply a subset of process and project metrics to support project management (a, i, k)
4. Construct a software project schedule and track its progress (a, i, k)
5. Perform simple estimates of software cost and level of effort for an object-oriented project (a, i, k)
6. Define common software quality assurance activities (a, i, k)
7. Conduct a formal technical review (f, k)
8. Define common elements of a software configuration management (SCM) system (a, i, k)
9. Utilize an SCM repository for project artifacts (a, i, k)
10. Apply software engineering principles and skills to complete a team-oriented software project (a, b, c, d, i, k)
11. Independently investigate a software engineering topic and prepare a suitable presentation (a, f)
12. Independently investigate an ethical or professional conduct topic and prepare a suitable presentation (e, f, g, h)

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•								•		•
2	•								•		•
3	•								•		•
4	•								•		•
5	•								•		•
6	•								•		•
7						•					•
8	•								•		•
9	•								•		•
10	•	•	•	•					•		•
11	•					•					
12					•	•	•	•			

- Program
- a. An ability computing to the
  - b. An ability identify
  - c. its
  - d. An ability evaluate a process, meet
  - e. An ability teams to
  - f. An security
  - g. An ability with a range of audiences.
  - h. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
  - i. Recognition of the need for and an ability to engage in continuing professional development.
  - j. An ability to use current techniques, skills, and tools necessary for computing practice.
  - k. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
  - l. An ability to apply design and development principles in the construction of software systems of varying complexity.

- Outcomes
- to apply knowledge of and mathematics appropriate discipline.
  - to analyze a problem, and and define the computing requirements appropriate to solution.
  - to design, implement, and computer-based system, component, or program to desired needs.
  - to function effectively on accomplish a common goal.
  - understanding of professional, ethical, legal, and social issues and responsibilities.
  - to communicate effectively

Prerequisites by Topic

1. Principles of Software Engineering
2. Software Development Life-Cycle
3. Team-oriented Software Development
4. Software Requirements and Specification
5. Software Design
6. Software Construction
7. Software Validation and Verification

Major Topics Covered in the Course

1. Product and Process Metrics
2. Project Management

3. Software Cost Estimation
4. Configuration Management
5. Software Quality Assurance
6. Software Modeling
7. Professionalism

Assessment Plan for the Course

The course utilizes two tools for assessment: Student performance on a large group project, typically solicited from an external client, and the results of a student survey. This course requirement provides an objective measure of level-of-achievement for course outcomes. The project is evaluated by both the instructor and client(s). The survey asks students to judge how well they achieved each learning outcome on a 5-pt Likert scale. The instructor is given the results and encouraged to make recommendations for the next offering, especially for those course outcomes whose average objective or subjective scores fall below 3.

How Data in the Course is Used to Assess Program Outcomes (unless adequately covered already in the assessment discussion under Criterion 4)

The results of the instructor and client project evaluation and student survey are the principal means by which program outcomes are assessed.

Estimate Curriculum Category Content (Semester hours)

Area	Core	Advanced	Area	Core	Advanced
Algorithms			Software design		4
Data structures			Concepts of programming languages		
Computer organization and architecture					

## COURSE DESCRIPTION

Dept. & Course Number : CS 472  
Semester Hours: 3.0

Course Title: Operating Systems Design  
Course Coordinator: Mark C. Temte

### Current Catalog Description

The design and implementation of modern multiprocessing operating systems. Topics include concurrent programming, real and virtual storage allocation, resource allocation and deadlock prevention and avoidance, job scheduling, and analytic modeling. Students will complete projects involving concurrency and implement a portion of a multiprocessing operating system.

### Textbook

W. Stallings, *Operating Systems: Internals and Design Principles (6th edition)*, Pearson Prentice-Hall (2009). ISBN 0-13-600632-9.

### References

None.

### Course Outcomes

Upon successful completion of the course requirements, a student should be able to:

1. Describe the basic resource management responsibilities of an operating system (a).
2. Describe the concept of a process and list the various process state transitions (a).
3. Distinguish between a process and a thread (a, j).
4. Analyze a concurrent programming application and apply appropriate techniques to avoid control problems: mutual exclusion, deadlock, and starvation (b, c, i, k).
5. Describe or apply virtual memory concepts (a, b, j).
6. Describe scheduling policies appropriate for both uniprocessor and multiprocessor systems (a, i, j, k).
7. Describe or apply the various disk scheduling techniques (a, b, i, j, k).
8. Describe or apply basic algorithms associated with distributed process management (a, b, i, k).
9. Design and implement a concurrent programming application using only semaphores for process control (b, c, i, k).
10. Design and implement a concurrent programming application in the Ada language using the message-passing protocol with advanced control features (b, c, i, k).
11. Design and implement a concurrent programming application using a monitor for process control (b, c, i, k).
12. Design and implement a simulation of operating an operating system feature (b, c, j, k).

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•										
2	•										
3	•									•	
4		•	•						•		•
5	•	•								•	
6	•								•	•	•
7	•	•							•	•	•
8	•	•							•		•
9		•	•						•		•
10		•	•						•		•
11		•	•						•		•
12		•	•							•	•

### Program Outcomes

- An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- An ability to function effectively on teams to accomplish a common goal.
- An understanding of professional, ethical, legal, security and social issues and responsibilities.
- An ability to communicate effectively with a range of audiences.
- An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- Recognition of the need for and an ability to engage in continuing professional development.
- An ability to use current techniques, skills, and tools necessary for computing practice.
- An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- An ability to apply design and development principles in the construction of software systems of varying complexity.

### Prerequisites by Topic

- Understanding of processor, bus, and memory operation
- Knowledge of virtual memory and cache memory
- Familiarity with interrupts and traps
- Understanding of I/O devices, techniques, and DMA
- Familiarity with multiprocessor and multicomputer architectures
- Understanding of the process concept
- Programming experience in a high-level language such as Java
- Knowledge of basic data structures and abstract data types

### Major Topics Covered in the Course (course hours)

Review of computer system concepts (1 hour)  
 Evolution of operating systems (3 hour)  
 Process concepts and management (4 hours)  
 Threads (1 hour)  
 Symmetric multiprocessing (SMP) (1 hour)  
 Microkernel architecture (1 hour)  
 Concurrent programming, mutual exclusion, synchronization, and deadlock (13 hours)  
 Memory management (2 hours)  
 Virtual memory and page replacement algorithms (3 hours)  
 Uniprocessor scheduling (2 hours)  
 Multiprocessor and real-time scheduling (2 hours)  
 File system management and disk scheduling (4 hours)  
 Distributed processing, client/server, and clusters (2 hours)  
 Distributed process management, global system state, Lamport's timestamp (3 hours)  
 Tests (4)

### Assessment Plan for the Course

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

### How Data in the Course is Used to Assess Program Outcomes

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 472, Program Outcomes a, b, c, i, j, and k are supported.

### Estimate Curriculum Category Content

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0.5	0.5	Software design	0.5	0.5
Data structures	0	0	Concepts of programming languages	0	0.5
Computer organization and architecture	0	0.5			

## COURSE DESCRIPTION

Dept. & Course Number : CS 474  
Semester Hours: 3.0

Course Title: Compiler Construction  
Course Coordinator: Mark C. Temte

### Current Catalog Description

Techniques for the syntax-directed translation of modern high-level languages. Topics include grammars and language specification, language design issues, lexical analysis, LL and LR parsing techniques, semantics, symbol table design, code generation, and local optimization. Students are required to implement a compiler for a subset of a structured high-level language such as Pascal or Ada.

### Textbook

Fischer and LeBlanc, *Crafting a Compiler with C*, Benjamin/Cummings (1991). ISBN 0-8053-2166-7.

### References

Mark Temte, *Project Implementation Notes for CS 474 (Compiler Construction)*, an unpublished 61 page manual prepared to guide students through the semester project.

### Course Outcomes

Upon successful completion of the course requirements, a student should be able to:

1. Understand elements of lexical analysis, recursive-descent parsing, and syntax-directed translation (a, i).
2. Calculate First and Follow sets for a grammar and do grammar analysis (a, b, i).
3. Modify a context-free grammar to eliminate common prefixes and left-recursion (a, b, i).
4. Use a grammar analysis tool to generate a parse table (a, b, i).
5. Demonstrate facility with the LL(1) parsing technique (a, i).
6. Understand LR and SLR(1) parsing (a, i).
7. Design and implement a compilation environment for Smalltalk that allows incremental compilation of new classes and methods (a, b, c, d, i, j, k).
8. Design and implement a scanner for Smalltalk (a, b, c, d, i, k).
9. Write semantic action routines for a compiler that performs code generation (a, b, c, d, i, j, k).
10. Design and implement a working Smalltalk compiler that works with a compilation environment (a, b, c, d, i, j, k).

## Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•								•		
2	•	•							•		
3	•	•							•		
4	•	•							•		
5	•								•		
6	•								•		
7	•	•	•	•					•	•	•
8	•	•	•	•					•		•
9	•	•	•	•					•	•	•
10	•	•	•	•					•	•	•

## Program Outcomes

- a. An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- c. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- d. An ability to function effectively on teams to accomplish a common goal.
- e. An understanding of professional, ethical, legal, security and social issues and responsibilities.
- f. An ability to communicate effectively with a range of audiences.
- g. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- h. Recognition of the need for and an ability to engage in continuing professional development.
- i. An ability to use current techniques, skills, and tools necessary for computing practice.
- j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- k. An ability to apply design and development principles in the construction of software systems of varying complexity.

## Prerequisites by Topic

- Ability to implement a medium-size program
- Knowledge of basic data structures and abstract data types
- Knowledge of statement-level control structures and parameter passing methods
- Basics of finite automata
- Understanding of formal languages, BNF, syntax, and semantics
- Knowledge of recursive descent parsing

## Major Topics Covered in the Course (course hours)

Introduction to compilers (1 hour)  
Survey of Smalltalk, the compilation environment, and the virtual machine (4 hours)  
Finite automata and lexical analysis (4 hours)  
Context free grammar design and grammar analysis algorithms (8 hours)  
LL(1) grammars and parsing (8 hours)  
LR parsing (6 hours)  
Symbol tables (2 hours)  
Run-time storage organization (2 hours)  
Code generation and optimization (9 hours)  
Tests (2 hours)

## Assessment Plan for the Course

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

## How Data in the Course is Used to Assess Program Outcomes

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 474, Program Outcomes a, b, c, d, i, j, and k are supported.

## Estimate Curriculum Category Content

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0	0.5	Software design	0	1.5
Data structures	0	0.5	Concepts of programming languages	0	0.5
Computer organization and architecture	0	0			

## COURSE DESCRIPTION

Dept. & Course Number	CS 486	Course Title	Analysis of Algorithm
Semester Hours	3.0	Course Coordinator	Peter A. Ng

### Current Catalog Description

Techniques for analyzing the time and space requirements of algorithms and problems. Application of these techniques to sorting, searching, pattern-matching, graph problems, and other selected problems. Brief introduction to the intractable (NP-hard) problems.

### Textbook

Anany Levitin, Introduction to *The Design & Analysis of Algorithms* (2<sup>nd</sup> Edition), Addison Wesley 2007, ISBN 0-321-35828-7.

### References

*Introduction to Algorithms*, Cormen, Leiserson, Rivest and Steain, MIT Press/McGraw Hill.  
*The Art of computer Programming* (2<sup>nd</sup> Edition) (3 volumes), Donald Knuth, Addison-Wesley.

### Course Outcomes

Upon successful completion of the course requirements, a student should be able to:

1. Develop a broad toolkit of most of standard algorithms for their common uses. (a, i, k).
2. Analyze the expected performance of a particular algorithm in a particular context. (b, j).
3. Utilize mathematical techniques to analyze the efficiency of an algorithm and demonstrate that the algorithm is correct – that it does in fact what it is claimed to do. (a, i, j).
4. Vary familiar algorithms, and devise new ones, to cope with unfamiliar contexts. (a, b, c, i, j, k).
5. Use the current techniques and skills necessary for analyzing and designing programs (algorithms) with complexity. (i).
6. Apply algorithmic principles with applicable mathematical techniques in the analysis and design of programs in a way that demonstrates comprehension of the tradeoffs involved in design and complexity choices. (j).
7. Investigate issues of ethics and professionalism of the use and design of algorithms during program construction. (e).
8. Apply analysis and design of algorithms in the construction of programs with varying complexity. (k).

## Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•								•		•
2		•								•	
3	•								•	•	
4	•	•	•						•	•	•
5									•		
6										•	
7					•						
8											•

## Program Outcomes

- An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- An ability to function effectively on teams to accomplish a common goal.
- An understanding of professional, ethical, legal, security and social issues and responsibilities.
- An ability to communicate effectively with a range of audiences.
- An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- Recognition of the need for and an ability to engage in continuing professional development.
- An ability to use current techniques, skills, and tools necessary for computing practice.
- An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- An ability to apply design and development principles in the construction of software systems of varying complexity.

## Prerequisites by Topic

- Data Structures
- Discrete Mathematics
- Calculus

## Major Topics Covered in the Course

- Introduction (1 hr.)
- Analysis framework and asymptotical notations (3 hrs.)
- Mathematical analysis of non-recursive algorithms ((1 hr.)
  - Algorithms MaxElement, UniqueElement, MatrixMultiplications, Binary(n)

- Mathematical analysis of recursive algorithms (2 hrs.)
  - Algorithms BinRec(n), Factorial F(n), HanoiTower(n), FibonacciNos F(n), FibonacciNos Fib(n)
- Brute Force (1 hr.)
  - Algorithms: SelectionSort, BubbleSort, SequentialSearch, BruceForceStringMatch
- Backward Substitution and Master Theorem (1 hr.)
- Divide-and-Conquer (2 hrs)
  - Mergesort, QuickSort, BinarySearch
- Other Divide-and-Conquer (1 hr )
  - Binary Tree Traversals (4.4), Strassen's Matrix Multiplication (4.5)
- Midterm Exam and Study for Exam (3 hrs.)
- Introduction to Methods for Solving Recurrence Relations: (1 hr.)
  - Method of Forward and Forward Substitutions
- Decrease-and Conquer (6 hrs.)
  - Decrease-by-One: Insertion Sort, DFS and BFS, Topological Sorting (5.1 – 5.3)
  - Decrease-by-Constant-Factor Algorithms (5.5)
  - Variable-size-decrease Algorithms) (5.6)
- Transform-and-Conquer (6 hrs.)
  - Instance-Simplification: Presorting, Gaussian Elimination, Balanced Search Tree (6.1 – 6.3).
  - Representation Change: Heaps and Heapsort or Horner's rule and Binary Exponentiation Problem Reduction (6.5)
- Space and Time Tradeoffs: (3 hrs.)
  - String Matching, Hashing, B-Trees (7.2 – 7.4)
- Dynamic Programming Algorithms (3 hrs.)
  - Binomial Coefficient Computation, Floyd's Algorithm, Optimal Binary Search Trees, Knapsack Problem (8.1 – 8.4)
- Greedy Algorithms: (3 hrs.)
  - Prim's, Krushal's Dijkstra's , Haffman's (9.1 – 9.4)
- P, NP, and NP-Complete Problems (11.3) (1 hr.)
- Coping with the Limitations of Algorithm Power (3 hrs.)
  - Backtracking: n-Queens Problem, Branch-and-Bound: Knapsack Problem, Traveling Salesman Problem and Assignment Problem
  - Approximation Algorithms: Traveling Salesman and Knapsack Problems.
- Final Examinations and Study for Exam (4 hrs.)

### **Assessment Plan for the Course**

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

### **How Data in the Course is Used to Assess Program Outcomes (unless adequately covered already in the assessment discussion under Criterion 4)**

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 486, Program Outcomes a, b, c, e, i, j, and k are supported.

**Estimate Curriculum Category Content (Semester hours)**

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0	2.5	Software design	0	0
Data structures	0	0.5	Concepts of programming languages	0	0
Computer organization and architecture	0	0			

## COURSE DESCRIPTION

Dept. & Course Number	CS 488	Course Title	Theory of Computation
Semester Hours	3.0	Course Coordinator	Peter A. Ng

### Current Catalog Description

Mathematical models of computation including finite and pushdown automata and Turing machines and equivalence of different general-purpose models. Grammars and their relation to automata, Church's Thesis, and limits of computation.

### Textbook

Cohen, Daniel I. A., *Introduction to Computer Theory*, Second Edition, John Wiley & Sons, 1996, ISBN 978-0-471-13772-6

### References

Hopcroft, J.E., Motwani, R. and J.D. Ullman, *Introduction to Automata Theory Languages and Computation, Third Edition*, Addison-Wesley, 2007, ISBN 10-0-321-46225-4.

### Course Outcomes

Upon successful completion of the course requirements, a student should be able to:

1. Comprehend some abstract models of the process of computation such as finite automata, pushdown automata and Turing machines.(a, j)
2. Follow basic mathematical arguments couched in terms of these models. (a, j)
3. Understand the basic theory of language recognition motivated by applications in areas such as compiler design. (a, i, j)
4. Solve problems and produce reasoned arguments about the power of the computational models studied in the course; and (a, b, c, f, j, k)
5. Write such arguments clearly and correctly with a proper use of mathematical notation where appropriate.(a, b, f, j)
6. Use their understanding of these models to break down and analyze the problems.(a, b, j)
7. Learn a range of comprehension, writing and problem-solving skills.(a, b, j)
8. Investigate which problems can be solved by the computer and which problems cannot be solved by computer, or, at least, by machines corresponding to the mathematical models of computers. (a, b, c, j, k)

## Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1	•									•	
2	•									•	
3	•								•	•	
4	•	•	•			•				•	•
5	•	•				•				•	
6	•	•								•	
7	•	•								•	
8	•	•	•							•	•

## Program Outcomes

- An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- An ability to function effectively on teams to accomplish a common goal.
- An understanding of professional, ethical, legal, security and social issues and responsibilities.
- An ability to communicate effectively with a range of audiences.
- An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- Recognition of the need for and an ability to engage in continuing professional development.
- An ability to use current techniques, skills, and tools necessary for computing practice.
- An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- An ability to apply design and development principles in the construction of software systems of varying complexity.

## Prerequisites by Topic

- Discrete Mathematics
- Language Evaluation Criteria; Language Design Trade-off
- Introduction to Formal Languages; Syntax and Semantics; Lexical and Syntax Analysis
- Names, Bindings, Scopes; Data Types

## Major Topics Covered in the Course

- Automata Theory (16 hours)
  - Background, Languages, Recursive Definitions, Regular Expressions, Finite Automata, Transition Graphs, Kleene's Theorem, Finite Automata with Output
  - Language Acceptors, Regular Grammars and their Languages, Equivalence of Regular Grammars and Finite Automata, the Concepts of Determinism and Non-Determinism, the Pumping Lemma for Regular Languages, Non-Regular Languages, Decidability.
- Pushdown Automata Theory (12 hours)

- Pushdown Automata and Context-Free Grammars and their Languages, Grammatical Format, Pushdown Automata, Equivalence of Pushdown automata and Context-Free Grammars, Non-Context-Free Languages, Decidability
- Turing Theory (9 hours)
  - Turing Machines(TM), Halting Problem of TM, Post Machines, Minsky's Theorem, Church's Thesis, Extension of Turing Machines, Turing Machine Languages and their Type-0 Grammars, The Chomsky Hierarchy, Computers.
- Complexity (2 hours)
  - Space and Time complexity and their relationship between them. Determinism versus Non-Determinism, P and NP, Reductions and Completeness.
- Tests (6 hours)

**Laboratory Projects (Specify Number of Weeks on Each)**

1. Develop a program to permit an end user to specify a finite automaton and run sample inputs strings, testing for acceptance (3 weeks)
2. Develop a program to permit an end user to specify a deterministic pushdown automaton and run sample input strings, testing for acceptance (3 weeks)
3. Develop a program to permit an end user to specify a deterministic Turing machine and run sample input strings, testing for acceptance (3 weeks)

**Assessment Plan for the Course**

Each time the course is offered, the class is initially informed of the list of Course Outcomes, which are included in the syllabus. Then, at the end of the semester, an anonymous survey of the class is conducted. For each Course Outcome, each student is asked to judge how well the outcome was achieved. The choices available are the following: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly Disagree". Subsequently, the results are converted to a 5-point scale and tabulated by the department, together with the average for each question. Once the data from the survey are tabulated, the results are returned to the instructor of the course. The instructor then analyzes the results of the survey and makes written recommendations for better achieving the Course Outcomes the next time the course is offered.

**How Data in the Course is Used to Assess Program Outcomes (unless adequately covered already in the assessment discussion under Criterion 4)**

Each Course Outcome of the assessment survey directly supports one or more of the Program Outcomes (see "Relationship between Course Outcomes and Program Outcomes" above). For CS 488, Program Outcomes a, b, c, f, i, j, and k are supported.

**Estimate Curriculum Category Content (Semester hours)**

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0	2.0	Software design	0	0
Data structures	0	0	Concepts of programming languages	0	1.0
Computer organization and architecture	0	0			

## COURSE DESCRIPTION

Dept., Number	CS 492	Course Title	Rich Internet Applications
Semester hours	3	Course Coordinator	R. L. Sedlmeyer

### Current Catalog Description

P: 372 or equivalent knowledge. Introduction to technologies and programming techniques that enable construction of rich internet applications, i.e., those which provide a user experience similar to desktop applications. Review of Javascript. Exploration of Javascript libraries that support common client-side behaviors and aid integration of content from several sources (mashups). Overview of the Ajax programming model and its role in building more responsive web applications. Students will be required to investigate and present a non-trivial Javascript application and complete a term project.

### Textbook

Practical JavaScript, DOM Scripting and Ajax Projects, by Frank Zammetti, Apress (April 16, 2007), ISBN-10: 1590598164 , ISBN-13: 978-1590598160 .

### References

1. [JavaScript: The Definitive Guide](#), David Flanagan O'Reilly Media, Inc., 5th edition, 2006.
2. [Professional JavaScript for Web Developers \(Wrox Programmer to Programmer\)](#), Nicholas C. Zakas, Wrox 2nd edition, 2009.
3. [Head First JavaScript](#), Michael Morrison, O'Reilly Media, Inc., 2008.
4. Ajax: The Definitive Guide, [Anthony T. Holdener III](#), O'Reilly Media, Inc., 2008.
5. [www.w3schools.com](http://www.w3schools.com) tutorials on Javascript, AJAX, HTML DOM and XML DOM

### Course Outcomes

1. Construct object-oriented client-side logic using Javascript (c, i)
2. Implement common client-side functionality (validation, persistence, mashups, XML parsing, drag-and-drop) in Javascript (b, c, i)
3. Utilize Javascript libraries to construct rich web application interfaces (b, c, i)
4. Utilize Javascript libraries to construct cross-browser web application interfaces (b, c, i)
5. Independently investigate and describe components of a Javascript library (f, h)
6. Describe the Ajax programming model (a, c, i)
7. Implement a non-trivial web application that incorporates the Ajax programming model (b, c, i, k)

### Relationship between Course Outcomes and Program Outcomes

The numbered Course Outcomes support the Program Outcomes as indicated in the following table, where the Program Outcomes (a-k) are listed below the table:

Course Outcome	Program Outcomes										
	a	b	c	d	e	f	g	h	i	j	k
1			•						•		
2		•	•						•		
3		•	•						•		
4		•	•						•		
5						•		•			
6	•		•						•		
7		•	•						•		•

### Program Outcomes

- An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
- An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- An ability to function effectively on teams to accomplish a common goal.
- An understanding of professional, ethical, legal, security and social issues and responsibilities.
- An ability to communicate effectively with a range of audiences.
- An ability to analyze the local and global impact of computing on individuals, organizations, and society.
- Recognition of the need for and an ability to engage in continuing professional development.
- An ability to use current techniques, skills, and tools necessary for computing practice.
- An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- An ability to apply design and development principles in the construction of software systems of varying complexity.

### Prerequisites by Topic

- HTML
- CSS
- Java server-side technologies (servlets and JSPs)
- HTTP protocol
- Web-application architecture

### Major Topics Covered in the Course

- Javascript language features and programming techniques.
- Javascript libraries for supporting rich client interfaces.
- Ajax programming model.
- Representing data using JSON and XML.
- DOM scripting.

### Assessment Plan for the Course

The course utilizes three tools for assessment: Student performance on a chapter case study, student performance on a large semester project, and the results of a student survey. The case study and large semester project together cover all course outcomes and each covers a majority of them. These course requirements provide an objective measure of level-of-achievement for course outcomes. The survey asks students to judge how well they achieved each learning outcome on a 5-pt Likert scale. The instructor is given the results and encouraged to make recommendations for the next offering, especially for those course outcomes whose average objective or subjective scores fall below 3.

**How Data in the Course is Used to Assess Program Outcomes (unless adequately covered already in the assessment discussion under Criterion 4)**

Since this is an optional course, data are not directly used to assess program outcomes.

**Estimate Curriculum Category Content (Semester hours)**

Area	Core	Advanced	Area	Core	Advanced
Algorithms			Software design		2.5
Data structures			Concepts of programming languages		0.5
Computer organization and architecture					

### **COURSE DESCRIPTION**

Dept. & Course Number	CS 292	Course Title	Intermediate Topics in Computer Science
Semester Hours	2.0-3.0	Course Coordinator	Departmental Chair

#### **Current Catalog Description**

Intermediate seminar addressing current topics or issues in computer science or information systems.

## **COURSE DESCRIPTION**

Dept. & Course Number	CS 395	Course Title	Industrial Practice I
Semester Hours	0-3.0	Course Coordinator	Departmental Chair

### **Current Catalog Description**

Practical problems in local industry limited to about 10-20 hours per week. May be repeated, but the total combined credit that may be applied to a degree is limited to 6. Open only to full-time students. Permission of the department is required.

## COURSE DESCRIPTION

Dept. & Course Number	CS 494	Course Title	Directed Study
Semester Hours	1.0-3.0	Course Coordinator	Departmental Chair

### Current Catalog Description

Independent study for students who desire to execute a complete computer-oriented project. Course may be repeated for credit up to 6 hours toward graduation.

## **COURSE DESCRIPTION**

Dept. & Course Number	CS 495	Course Title	Cooperative Experience
Semester Hours	0-3.0	Course Coordinator	Departmental Chair

### **Current Catalog Description**

For Cooperative Education students only. Permission of the department required.