D. COMPUTER SCIENCE COURSES

COSC 2331: Discrete Structures
COSC 3343: Database Theory
COSC 3351: Algorithms
COSC 3411: System Programming
COSC 3421: Data Structures
COSC 4311: Parallel Computing
COSC 4361: Operating Systems
COSC 4362: Artificial Intelligence
COSC 4363: Automata Theory
COSC 4364: Compilers
COSC 4461: Programming Languages

Course Title: COSC 2331: Discrete Structures

Semester Credit Hours: 3 (3,0)

I. Course Overview

Discrete Structures is the study of objects that have discrete as opposed to continuous values including the foundations of logic, algorithms and their complexity, mathematical reasoning, relations, graphs, trees and combinatorics.

II. PMU Competencies and Learning Outcomes

Students of COSC 2331: Discrete Structures develop the quantitative skills necessary for continued success in computer science These skills enhance their ability to both analyze and describe mathematically many of the algorithms and data structure performance characteristics common to computer science as a discipline and to effectively communicate their solutions to fellow professionals. The course makes extensive use of the technology infrastructure of the school for communication within the class and between students and the instructor. Through the use of group tasks and projects this course establishes both mathematical reasoning skills and technical communication skills.

III. Detailed Course Description

COSC 2331: Discrete Structures is concerned with the application of objects with discrete characteristics to computer science as a discipline in order that commonly used structures may be described, characterized and analyzed. The course examines the fundamentals of propositional logic and set operations, the analysis of algorithm complexity, mathematical reasoning including proofs and induction, recursion, and program correctness. Graphs, including Euler and Hamilton Paths and shortest path problems are examined, as are tree applications. Te course concludes with an investigation of Boolean functions, gate representations and approaches to circuit minimization.

IV. Requirements Fulfilled

This course satisfies three hours of the requirements for the degree in computer science. It is required of all students pursuing the computer science degree from the College of Information Technology. It should be taken immediately after completion of MATH 1323: Calculus II.

V. Required Prerequisites

MATH 1423: Calculus II

VI. Learning Outcomes

In this course, students learn:

- To develop understanding of Logic Sets and Functions.
- To use mathematical reasoning techniques including induction and recursion
- To understand and apply counting techniques to the representation and characterization of relational concepts.
- To develop an understanding of how graph and tree concepts are used to solve problems arising in the computer science.
- To communicate the solutions of technical problems to other professionals.
- To develop improved collaborative skills.

VII. Assessment Strategy

This course is designed with three primary goals in mind: to further the student's understanding of mathematics, to lead students to connect the mathematics to its application in computer science and to encourage the students to communicate their ideas and their expertise to the professional community. With this in mind, the course grade involves an assessment of their performance on in-class quizzes and exams that focus on the applications of discrete mathematics to computer science.

- Weekly assigned homework to motivate students to do the work and earn credit accordingly.
- Weekly in-class quizzes
- Weekly, in-class presentations by students of solutions to real world problems related to the course material and classroom discussion and critique of the presentation.
- Three in-class exams to assess the student's accumulative mastery of content covered prior to time of exam.
- A comprehensive final exam to assess the student's accumulative mastery of course material.

The final grades is based on 15% credit for the homework, 15% for the quizzes, 10% for the presentations and participation in classroom discussion, 30% on in-class exams, and 30% for the final exam.

Students are required to maintain a journal of thoughts and commentaries during the course. The journal contains daily entries including the identification of areas of interest and concern, notes on the preparation of presentation and comments and analysis of classmate's presentations. The journal is reviewed weekly by the instructor to provide feedback to the students.

Final grades and the student and instructor observations from reflective notebooks are included in the student's portfolio for use in the final assessment capstone course. The intent is to document the student's maturation as he proceeds through the curriculum.

VIII. Course Format

A. Instruction

Primary instruction is a lecture format, with the course meeting three times per week for one hour each meeting. At least once per week students should be prepared to make presentation on a topic selected by the instructor and to take part in a discussion based on that presentation. Once a week students should have at least 30 minutes of collaborative problem solving activity.

B. Web supplement

Course home page (the university's Web tool, WebCt or Blackboard) should contain the following:

- Course syllabus
- Course assignments
- Keys to quizzes and exams (after students have completed them)
- Course calendar (an active utility)
- Course e-mail (an active utility)
- Course discussion list (an active utility)
- Students course marks.(an active utility)

Classroom Hours (3 hours per week) **Class:** 3
Lab: 0

IX. Topics to be Covered

A. Logic, sets, and functions
   1. Logic
   2. Propositional equivalences
   3. Predicates and quantifiers
   4. Sets and set operations
   5. Functions

B. Algorithms, the integers, and matrices
   1. Algorithms and their complexity
   2. The integers and division
   3. Integers and algorithms
   4. Applications of number theory
   5. Matrices

C. Mathematical reasoning
   1. Methods of proof
   2. Mathematical induction
   3. Recursive algorithms
   4. Program correctness

D. Counting
    1. The pigeonhole principle
    2. Permutations and combinations
    3. Discrete probability
    4. Probability theory
    5. Generalized permutations and combinations
    6. Recurrence relations
    7. Generating functions
E. Relations
    1. Relations and their properties
    2. N-ary relations and their applications
    3. Representing relations
    4. Closures of relations
    5. Equivalence relations
    6. Partial orderings
F. Graphs
    1. Graph terminology
    2. Representing graphs and graph isomorphism
    3. Connectivity
    4. Euler and Hamilton paths
    5. Shortest path problems
    6. Planar graphs
    7. Graph coloring
G. Trees
    1. Introduction to trees
    2. Applications of trees
    3. Tree traversal
    4. Trees and sorting
    5. Spanning trees
    6. Minimum spanning trees
H. Boolean algebra
    1. Boolean functions
    2. Representing Boolean functions
    3. Logic gates
    4. Minimization of circuits

X. Laboratory Exercises

This course does not require a separate lab.

XI. Technology Component

This course has no technology component other than use of the student's personal laptop computers as appropriate.

XII. Special Projects/Activities

Students are required to keep a "reflective notebook" in which, after each class, they enter their own assessments of what they learned, and what questions remain from the class. From each exercise set, each student selects one problem, which the student thinks best reflects the way the topic is used in a technical context. A detailed solution to the problem is included in the student's reflective notebook.

XIII. Textbooks and Teaching Aids

A. Required Textbook

Rosen, K. H. (1999) *Discrete Mathematics*. McGraw-Hill.
ISBN 0-07-289905-0

B. Alternative Textbooks

Hall, C., & O'Donnell J. (2000) *Discrete Mathematics Using a Computer*, Springer Verlag
ISBN 1-85-233089-9

Balakrishnan, V.K. (1996). *Introductory Discrete Mathematics*, Dover
ISBN 0-48-669115-2

C. Supplemental Print Materials

As available from publisher.

D. Supplemental Online Materials

As available from publisher.

Course Title: **COSC 3343: Database Theory**

Semester Credit Hours: 3 (3,0)

I. Course Overview

This course is the study of the principles of database systems. The goal is to prepare students an understanding of the theory as well as practices of database management systems.

II. PMU Competencies and Learning Outcomes

Students of COSC 3343: Database Theory develop conceptual, quantitative, and programming skills necessary for continued success in computer science. The skills enhance their abilities to appreciate the theory and practices of database management systems common to computer science as a discipline and to effectively communicate their solutions to fellow professionals. This course makes extensive use of the PMU technology infrastructure to provide communication between faculty and students. The course includes individual as well as group projects, establishes both conceptual reasoning skills and technical communication skills, and provides opportunities for the presentation and defense of designed solutions.

III. Detailed Course Description

This course is concerned with the study of the principles of database systems, including conceptual modeling and data models, logical and physical database design, query languages and query processing, and database services such as concurrency control, crash recovery, security, and integrity.

IV. Requirements Fulfilled

COSC 3343: Database Theory satisfies three hours of the requirements for the degree in computer science. It is required of all students pursuing a degree program in computer science within the College of Information Technology. It should be taken in the first semester of the junior year.

V. Required Prerequisites

GEIT 1412: Computer Science II
MATH 1313: Statistical Methods

VI. Learning Outcomes

In this course, students learn:

- To understand the underlying design principles of database systems.
- To understand and apply the conceptual modeling and data models for database design.
- To understand the concepts on logical and physical database design.
- To understand and apply query languages and query processing.
- To understand the theory behind the database services.
- To develop improved communication and collaborative skills.

VII. Assessment Strategy

This course is designed with three primary goals in mind: to introduce students to the conceptual basis and practical issues associated with the database design, to lead students to connect the quantitative techniques to its application in computer science, and to provide students with the opportunity to communicate their designs and implementations to their peers in a professional setting. With this in mind, the course grade involves an assessment of their performance on examinations that focus on the application of programming paradigms to the solutions of problems, the performance analysis of the designed solutions, and the communication of designed solutions to those problems to an audience. Course grades are based on:

- Weekly assigned homework to motivate students to do the work and earn credit accordingly.
- Weekly, in-class presentations by students of solutions to real world problems related to the course material and classroom discussion and critique of the presentation.
- Two in-class examinations to assess the student's accumulative mastery of content covered prior to the time of the examination.
- Three programming assignments testing students understanding of the major concepts introduced during the course.
- A comprehensive final examination to assess the student's accumulative mastery of course material.

The final grades is based on 10% credit for the homework, 10% for the presentations and participation in classroom discussion, 30% on in-class examinations, 30% on programming assignments and 20% for the final examination.

Students are required to maintain a journal of thoughts and commentaries during the course. The journal contains daily entries including the identification of areas of interest and concern, notes on the preparation of presentation and comments and analysis of classmate's presentations. The journal is reviewed weekly by the instructor to provide feedback to the students.

Final grades and the student and instructor observations from reflective notebooks are included in the student's portfolio for use in the final assessment capstone course. The intent is to document the student's maturation as he proceeds through the curriculum.

VIII. Course Format

A. Instruction

This course utilizes both lecture/discussion and laboratory exercises. Students are expected to attend three hours of lecture per week. At least once per week students should be prepared to make presentation on the design and implementation of a solution to a problem selected by the instructor and to take part in a discussion based on that presentation. Once a week students should have at least 30 minutes of collaborative problem solving activity.

B. Web supplement

Course home page (the university's Web tool, WebCT or Blackboard) should contain the following:

- Course syllabus
- Course assignments
- Sample solutions to examinations (after being graded and returned)
- Sample solutions to programming assignments (after being graded and returned)
- Course calendar (an active utility)
- Course e-mail (an active utility)
- Course discussion list (an active utility)
- Student course performance (an active utility)

Classroom Hours (3 hours per week) **Class:** 3
Lab: 0

IX. Topics to Be Covered

A. Database system components
B. Entry-Relationship (ER) model
C. Relational model and relational algebra
D. Structured Query Language (SQL)
E. Relationship database design
F. ER to relationship mapping
G. Network model
H. Hierarchical model
I. Object-oriented model
J. Object-relational model
K. File organization
L. Index structures for files

M. Query processing and optimization
N. Transactions
O. Concurrency control
P. Crash recovery
Q. Security

X. Laboratory Exercises

This course does not offer a separate laboratory to students.

XI. Technology Component

This course makes use of the university's wireless access infrastructure. The course relies on the university and the students having access to professional grade application development environments for the students to use.

XII. Special Projects / Activities

Students are required to keep a "reflective notebook" in which, after each class, they enter their own assessments of what they learned, and what questions remain from the class. From each exercise set, each student selects one problem, which the student thinks best reflects the way the topic is used in a technical context. A detailed solution to the problem is included in the student's reflective notebook.

XIII. Textbooks and Teaching Aids

A. Required Textbook

Silberschatz, H. F. Korth, and S. Sudarshan. *Database System Concepts*. _____: McGraw-Hill, 2001.
ISBN 0-07-255481-9

B. Alternative Textbooks

Elmasri, R., and S. B. Navathe. *Fundamentals of Database Systems*. _____: Addison-Wesley, 2003.
ISBN 0-201-74153-9

C. Supplemental Print Materials

None.

D. Supplemental Online Materials

As available from the publishers.

Course Title: COSC 3351: Algorithms

Semester Credit Hours: 3 (3,0)

I. Course Overview

This course is the study of the design and performance analysis of algorithms. Time and space complexity analysis of algorithms, design paradigms, and graph algorithms are discussed.

II. PMU Competencies and Learning Outcomes

Students of COSC 3351: Algorithms develop quantitative and programming skills necessary for continued success in computer science. The skills enhance their abilities to devise, analyze, and comprehend mathematically the performance characteristics of algorithms and various design paradigms common to computer science as a discipline and to effectively communicate their solutions to fellow professionals. This course makes extensive use of the PMU technology infrastructure to provide communication between faculty and students. The course includes individual as well as group projects, establishes both mathematical reasoning skills and technical communication skills, and provides opportunities for the presentation and defense of designed solutions.

III. Detailed Course Description

This course is concerned with the study of the design and performance analysis of algorithms, including maximum contiguous subarray, divide-and-conquer algorithms, graph algorithms, dynamic programming, and greedy algorithms. One important lasting effect of this course is to enhance and develop the ability to specify, design, implement, test, and analyze solutions to programming problems utilizing the proven design paradigms presented in this course.

IV. Requirements Fulfilled

COSC 3351: Algorithms satisfies 3 hours of the requirements for the degree in computer science. It is required of all students pursuing a degree program in computer science within the College of Information Technology. It should be taken in the second semester of the junior year.

V. Required Prerequisites

- COSC 3421: Data Structures
- MATH 1313: Statistical Methods

VI. Learning Outcomes

In this course, students learn:

- To understand and apply performance analysis techniques to analyze maximum contiguous subarray.
- To understand and analyze the performance of divide-and-conquer algorithms.
- To understand and analyze the performance of graph algorithms.
- To understand and analyze the performance of dynamic programming.
- To understand and analyze the performance of greedy algorithms.
- To understand the concepts of complexity classes.
- To use mathematical and measurement techniques to analyze the performance characteristics of algorithms.
- To understand and apply techniques of algorithm analysis and express performance characteristics of algorithms.
- To develop improved communication and collaborative skills.

VII. Assessment Strategy

This course is designed with three primary goals in mind: to introduce students to the conceptual basis and practical issues associated with the use, development, and analysis of designed algorithms, to lead students to connect the mathematics to its application in computer science, and to provide students with the opportunity to communicate their designs and implementations to their peers in a professional setting. With this in mind, the course grade involves an assessment of their performance on examinations that focus on the application of programming paradigms to the solutions of problems, the performance analysis of the designed solutions, and the communication of designed solutions to those problems to an audience. Course grades are based on:

- Weekly assigned homework to motivate students to do the work and earn credit accordingly.
- Weekly, in-class presentations by students of solutions to real world problems related to the course material and classroom discussion and critique of the presentation.
- Two in-class examinations to assess the student's accumulative mastery of content covered prior to the time of the examination.
- Three programming assignments testing students understanding of the major concepts introduced during the course.
- A comprehensive final examination to assess the student's accumulative mastery of course material.

The final grade is based on 10% credit for the homework, 10% for the presentations and participation in classroom discussion, 30% on in-class examinations, 30% on programming assignments and 20% for the final examination.

Students are required to maintain a journal of thoughts and commentaries during the course. The journal contains daily entries including the identification of areas of interest and concern, notes on the preparation of presentation and comments and analysis of classmate's presentations. The journal is reviewed weekly by the instructor to provide feedback to the students.

Final grades and the student and instructor observations from reflective notebooks are included in the student's portfolio for use in the final assessment capstone course. The intent is to document the student's maturation as he proceeds through the curriculum.

VIII. Course Format

A. Instruction

This course utilizes both lecture/discussion and laboratory exercises. Students are expected to attend three hours of lecture per week. At least once per week students should be prepared to make presentation on the design and implementation of a solution to a problem selected by the instructor and to take part in a discussion based on that presentation. Once a week students should have at least 30 minutes of collaborative problem solving activity.

B. Web supplement

Course home page (the university's Web tool, WebCT or Blackboard) should contain the following:

- Course syllabus
- Course assignments
- Sample solutions to examinations (after being graded and returned)
- Sample solutions to programming assignments (after being graded and returned)
- Course calendar (an active utility)
- Course e-mail (an active utility)
- Course discussion list (an active utility)
- Student course performance (an active utility)

IX. Topics to be Covered

A. Case study in algorithm design
   maximum contiguous subarray
B. Divide-and-conquer algorithms
   1. Merge sort
   2. Polynomial multiplication
   3. Randomized selection
C. Graphs
   1. Notations
   2. Breadth-first search and depth-first search
   3. Cycle finding and topological sort
   4. Maximum spanning trees
   5. Dijkstra's shortest path algorithm

D. Dynamic programming
    1. Knapsack
    2. Chain matrix multiplication
    3. Longest common subsequence
    4. All pairs shortest path
E. Greedy algorithms
    1. Activity selection
    2. Huffman coding
F. Complexity classes
    1. Nondeterminism
    2. Classes P and NP
    3. NP-complete problems
    4. Polynomial reductions

X. Laboratory Exercises

This course does not offer a separate laboratory to students.

XI. Technology Component

This course makes use of the university's wireless access infrastructure. The course relies on the university and the students having access to professional grade application development environments for the students to use.

XII. Special Projects/Activities

Students are required to keep a "reflective notebook" in which, after each class, they enter their own assessments of what they learned, and what questions remain from the class. From each exercise set, each student selects one problem, which the student thinks best reflects the way the topic is used in a technical context. A detailed solution to the problem is included in the student's reflective notebook.

XIII. Textbooks and Teaching Aids

A. Required Textbook

Cormen, T., C., R. Rivest Leiserson, and C. Stein. *Introduction to Algorithms*. _____: The MIT Press, 2001.
ISBN 0-262-03293-7

B. Alternative Textbooks

1. Bentley, J. *Programming Pearls*. _____: Addison-Wesley, 2000.
   ISBN 0-201-65788-0
2. Garey, M.R., and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, 1979.
   ISBN 0-7167-1045-5

C. Supplemental Print Materials

None.

D. Supplemental Online Materials

As available from publishers.

Course Title: COSC 3411: Systems Programming

Semester Credit Hours: 4 (3,1)

I. Course Overview

Systems programming is the study of the basic programming principles and skills for building systems software, including the introduction to UNIX, shell programming, and Perl programming.

II. PMU Competencies and Learning Outcomes

Students in this course develop skills necessary for building systems software over UNIX platform. These skills are necessary for continued success in computer science. This course makes extensive use of the PMU technology infrastructure to provide communication between faculty and students. The course includes a structured laboratory component to ensure that students gain the necessary experience and skill in managing the concepts introduced in the class. The course includes individual as well as group projects and provides opportunities for the presentation and defense of designed solutions.

III. Detailed Course Description

COSC 3411: Systems Programming is concerned with the basic programming principles and skills for building systems software, including the introduction to UNIX, shell programming, and Perl programming. The course presents the students with the concepts of UNIX editor, utilities, file systems, links and shells, shell programming, and Perl programming. Students are exposed to a variety of techniques for the implementation and uses of systems software. One important lasting effect of this course is to enhance and develop the ability to specify, design, implement and test solutions to programming problems utilizing the data structures and proven algorithms presented in this course.

IV. Requirements Fulfilled

This course satisfies four hours of the requirements for the degree in computer science. It is required of all students pursuing a degree program in computer science within the College of Information Technology. It should be taken in the first semester of the junior year.

V. Required Prerequisites

- GEIT 1412: Computer Science II
- GEIT 1311: Computer Organization

VI. Learning Outcomes

In this course, students learn:

- To develop an understanding of UNIX platform including the use of editor, utilities, file systems, links, shells, shell programming, and Perl programming.
- To be able to discuss the advantages and disadvantages of different implementations of each of the data structures.
- To understand and apply techniques of algorithm analysis and express performance characteristics of algorithms.
- To develop improved communication and collaborative skills.

VII. Assessment Strategy

This course is designed with three primary goals in mind: to introduce students to the conceptual basis and practical issues associated with the use and development of systems software, to provide students with significant experience in the development of systems software within a profession development environment, and to provide students with the opportunity to communicate their designs and implementations to their peers in a professional setting. With this in mind, the course grade involves an assessment of their performance on examinations that focus on the application of programming techniques to the solutions of problems and the communication of designed solutions to those problems to an audience. Course grades are based on:

- Weekly assigned homework to motivate students to do the work and earn credit accordingly.
- Weekly, in-class presentations by students of solutions to real world problems related to the course material and classroom discussion and critique of the presentation.
- Weekly structured laboratory exercises designed to guide students through specific course topics.
- Two in-class examinations to assess the student's accumulative mastery of content covered prior to the time of the examination.
- Five programming assignments testing students understanding of the major concepts introduced during the course.
- A comprehensive final examination to assess the student's accumulative mastery of course material.

The final grade is based on 10% credit for the homework, 10% for the presentations and participation in classroom discussion, 20% for weekly laboratory exercises, 20% on in-class examinations, 30% on programming assignments and 10% for the final examination.

Students are required to maintain a journal of thoughts and commentaries during the course. The journal contains daily entries including the identification of areas of interest and concern, notes on the preparation of presentation and comments and analysis of classmate's presentations. The journal is reviewed weekly by the instructor to provide feedback to the students.

Final grades and the student and instructor observations from reflective notebooks are included in the student's portfolio for use in the final assessment capstone course. The intent is to document the student's maturation as he proceeds through the curriculum.

VIII. Course Format

A. Instruction

This course utilizes both lecture/discussion and laboratory exercises. Students are expected to attend three hours of lecture/discussion per week and two hours of laboratory per week. At least once per week students should be prepared to make presentation on the design and implementation of a solution to a problem selected by the instructor and to take part in a discussion based on that presentation. Once a week students should have at least 30 minutes of collaborative problem solving activity.

B. Web supplement

Course home page (the university's Web tool, WebCT or Blackboard) should contain the following:

- Course syllabus
- Course assignments
- Sample solutions to examinations (after being graded and returned)
- Sample solutions to programming assignments (after being graded and returned)
- Course calendar (an active utility)
- Course e-mail (an active utility)
- Course discussion list (an active utility)
- Student course performance (an active utility)

Classroom Hours (5 hours per week) **Class:** 3
Lab: 2

IX. Topics to be Covered

A. UNIX environment
1. Editor
2. Utilities
3. File systems
4. Links and shells
5. Shell programming

B. Perl programming
   1. Basics
   2. Lists and arrays
   3. Control flow
   4. Subroutines
   5. Hashes
   6. Regular expressions
   7. File input and output

X. Laboratory Exercises

This course requires a weekly 2-hour laboratory component. Topics to be covered in the laboratory sessions should include:

- UNIX environment – exercises in getting start in UNIX environment.
- UNIX editor – exercises in familiarizing the use of an UNIX editor.
- UNIX utilities – exercises in using some UNIX utilities.
- UNIX file systems – exercises in access control in UNIX file systems.
- UNIX shells – exercises in extending the shell facilities in UNIX.
- Shell Programming I – exercises in developing shell programs over an UNIX shell.
- Shell Programming II – additional shell programming exercises emphasizing the use of control flow and positional parameters.
- Perl lists and arrays – exercises in using Perl lists and arrays.
- Perl control flow – exercises in mastering the use of Perl control flow.
- Perl hashes – exercises in using Perl hashes.
- Regular expressions – exercises in mastering the use of regular expressions in string matching in Perl.
- Perl input and output – exercises in Perl file access.

Three additional lab sessions should be kept in reserve to allow the instructor to extend the more difficult laboratories for more than one session.

XI. Technology Component

This course makes use of the university's wireless access infrastructure. The course relies on the university and the students having access to professional grade application development environments for the students to use. The course has a laboratory component that would be best implemented in university provided laboratory space.

XII. Special Projects/Activities

Students are required to keep a "reflective notebook" in which, after each class, they enter their own assessments of what they learned, and what questions remain from the class. From each exercise set, each student selects one problem, which the student thinks best reflects the way the topic is used in a technical context. A detailed solution to the problem is included in the student's reflective notebook.

XIII. Textbooks and Teaching Aids

A. Required Textbook

1. Das, S. *Your UNIX: The Ultimate Guide*. _____: McGraw-Hill, 2001.
   ISBN 0-07-240500-7
2. Schwartz, R. L., and T. Phoenix. *Learning Perl*. ____: O'Reilly, 2001.
   ISBN 0-596-00132-0

B. Alternative Textbooks

None.

C. Supplemental Print Materials

None.

D. Supplemental Online Materials

As available from publishers.

Course Title: COSC 3421: Data Structures

Semester Credit Hours: 4 (3,1)

I. Course Overview

Data structures is the systematic study of some advanced data structures, including list, stack, queue, dictionary, and graph. Sorting and hashing algorithms and their associated computational costs are discussed. Algorithm analysis techniques are also investigated to provide a metric to measure the performance of an algorithm in question.

II. PMU Competencies and Learning Outcomes

Students in this course develop programming and quantitative skills necessary for continued success in computer science. The skills enhance their abilities to devise, analyze, and comprehend mathematically the performance characteristics of algorithms and data structures common to computer science as a discipline and to effectively communicate their solutions to fellow professionals. This course makes extensive use of the PMU technology infrastructure to provide communication between faculty and students. The course includes a structured laboratory component to ensure that students gain the necessary experience and skill in managing the concepts introduced in the class. The course includes individual as well as group projects, establishes both mathematical reasoning skills and technical communication skills, and provides opportunities for the presentation and defense of designed solutions.

III. Detailed Course Description

COSC 3421: Data Structures is concerned with the systematic study of some advanced data structures, including list, stack, queues, dictionary, graphs. Sorting and hashing algorithms and their associated computational costs are discussed. The course presents the students with the concepts of asymptotic notations, performance measurement, sorting and searching including algorithms and lower bounds, abstract data types and classes, data structures such as heaps, search trees, tries, and hashing, and graphs: representation, depth-first-search, and breadth-first search. One important lasting effect of this course is to enhance and develop the ability to specify, design, implement, test, and analyze solutions to programming problems utilizing the data structures and proven algorithms presented in this course.

IV. Requirements Fulfilled

This course satisfies four hours of the requirements for the degree in computer science. It is required of all students pursuing a degree program in computer science within the College of Information Technology. It should be taken in the first semester of the junior year.

V. Required Prerequisites

- GEIT 1412: Computer Science II
- MATH 1313: Statistical Methods
- MATH 2332: Differential Equations

VI. Learning Outcomes

In this course, students learn:

- To understand and use of data structures, including list, stack, queue, dictionary, and graphs.
- To understand and use of sorting and hashing algorithms.
- Learn to use mathematical and measurement techniques to analyze the performance characteristics of algorithms.
- To be able to discuss the advantages and disadvantages of different implementations of each of the data structures.
- To understand and apply techniques of algorithm analysis and express performance characteristics of algorithms.
- To develop improved communication and collaborative skills.

VII. Assessment Strategy

This course is designed with three primary goals in mind: to introduce students to the conceptual basis and practical issues associated with the use, development, and analysis of data structures and algorithms, to lead students to connect the mathematics to its application in computer science, and to provide students with the opportunity to communicate their designs and implementations to their peers in a professional setting. With this in mind, the course grade involves an assessment of their performance on examinations that focus on the application of programming techniques to the solutions of problems, the performance analysis of the designed solutions, and the communication of designed solutions to those problems to an audience. Course grades are based on:

- Weekly assigned homework to motivate students to do the work and earn credit accordingly.
- Weekly, in-class presentations by students of solutions to real world problems related to the course material and classroom discussion and critique of the presentation.
- Weekly structured laboratory exercises designed to guide students through specific course topics.
- Two in-class examinations to assess the student's accumulative mastery of content covered prior to the time of the examination.
- Five programming assignments testing students understanding of the major concepts introduced during the course.
- A comprehensive final examination to assess the student's accumulative mastery of course material.

The final grade is based on 10% credit for the homework, 10% for the presentations and participation in classroom discussion, 20% for weekly laboratory exercises, 20% on in-class examinations, 30% on programming assignments and 10% for the final examination.

Students are required to maintain a journal of thoughts and commentaries during the course. The journal contains daily entries including the identification of areas of interest and concern, notes on the preparation of presentation and comments and analysis of classmate's presentations. The journal is reviewed weekly by the instructor to provide feedback to the students.

Final grades and the student and instructor observations from reflective notebooks are included in the student's portfolio for use in the final assessment capstone course. The intent is to document the student's maturation as he proceeds through the curriculum.

VIII. Course Format

A. Instruction

This course utilizes both lecture/discussion and laboratory exercises. Students are expected to attend three hours of lecture/discussion per week and two hours of laboratory per week. At least once per week students should be prepared to make presentation on the design and implementation of a solution to a problem selected by the instructor and to take part in a discussion based on that presentation. Once a week students should have at least 30 minutes of collaborative problem solving activity.

B. Web supplement

Course home page (the university's Web tool, WebCT or Blackboard) should contain the following:

- Course syllabus
- Course assignments
- Sample solutions to examinations (after being graded and returned)
- Sample solutions to programming assignments (after being graded and returned)
- Course calendar (an active utility)
- Course e-mail (an active utility)
- Course discussion list (an active utility)
- Student course performance (an active utility)

Classroom Hours (5 hours per week) **Class:** 3
Lab: 2

IX. Topics to be Covered

A. Basic data structures
   1. List
   2. Stack
   3. Queue

B. Analysis of algorithms
   1. Time complexity
   2. Upper and lower bounds

C. Sorting
   1. Insertion sort
   2. Merge sort
   3. Quick sort
   4. Heap sort
   5. Performance bounds

D. Dictionary
   1. Binary search trees
   2. AVL tree
   3. B tree

E. Graph
   1. Definition, representation, and modeling tool
   2. Breath-first search
   3. Depth-first search
   4. Directed graph and topological sort

F. Hashing
   1. Hash tables and functions
   2. Collision resolution

X. Laboratory Exercises

This course requires a weekly two-hour laboratory component. Topics to be covered in the laboratory sessions should include:

- Basic data structures – review exercises in basic data structures.
- Time complexity – exercises in estimating time complexity of algorithms by measurement.
- Performance bounds – exercises in estimating performance bounds of algorithms by measurement.
- Sorting I – exercises in the use and analyzing of sorting algorithms including merge sort and quick sort.
- Sorting II – additional exercises in the use and analyzing of sorting algorithms including heap sort.
- Dictionary I – exercises in the use of dictionary as an abstract data type including binary search tree and AVL tree.
- Dictionary II – additional exercises in the use of dictionary including B tree.
- Breath-first search – exercises in the use of breath-first search algorithm.
- Depth-first search – exercises in the use of depth-first search algorithm.
- Topological sort – exercises in the use of topological sort algorithm

- Hashing I – exercises in the implementation of hash tables and functions.
- Hashing II – exercises in the implementation of collision resolution.
- Three additional lab sessions should be kept in reserve to allow the instructor to extend the more difficult laboratories for more than one session.

XI. Technology Component

This course makes use of the university's wireless access infrastructure. The course relies on the university and the students having access to professional grade application development environments for the students to use. The course has a laboratory component that would be best implemented in university provided laboratory space.

XII. Special Projects/Activities

Students are required to keep a "reflective notebook" in which, after each class, they enter their own assessments of what they learned, and what questions remain from the class. From each exercise set, each student selects one problem, which the student thinks best reflects the way the topic is used in a technical context. A detailed solution to the problem is included in the student's reflective notebook.

XIII. Textbooks and Teaching Aids

A. Required Textbook

Weiss, M. A. *Data Structures and Algorithm Analysis in C++*. _____: Addison-Wesley, 1999.
ISBN 0-201-36122-1

B. Alternative Textbooks

Horowitz, E., S. Sahni, and D. Mehta. *Fundamentals of Data Structures in C++*. _____: W. H. Freeman, 1995.
ISBN 0-7167-8296-0

C. Supplemental Print Materials

None.

D. Supplemental Online Materials

As available from publishers.

Course Title: COSC 4311: Parallel Computing

Semester Credit Hours: 3 (3,0)

I. Course Overview

This course provides a basic, in-depth look at techniques for the design and analysis of parallel algorithms and for programming them on commercially available parallel platforms. Principles of parallel algorithms design and different parallel programming models are both discussed. MPI, POSIX threads, and Open MP all are discussed. This course is for anyone wanting to gain proficiency in all aspects of parallel and distributed programming.

II. PMU Competencies and Learning Outcomes

Students of COSC 4311: Parallel Computing develop skills necessary for understanding the design of parallel computing applications so as to appreciate the strengths and limitations of parallel computing approaches to problem solving. This course makes extensive use of the PMU technology infrastructure to provide communication between faculty and students. The course is primarily a lecture-based course with the student required to complete significant projects outside of class time. The course includes individual as well as group projects and provide opportunities for the presentation and defense of designed solutions. The course encourages the development of professional communication skills and provides opportunities for collaborative project development.

III. Detailed Course Description

This course provides a basic, in-depth look at techniques for the design and analysis of parallel algorithms and for programming them on commercially available parallel platforms. Principles of parallel algorithms design and different parallel programming models are both discussed. MPI, POSIX threads, and Open MP all are discussed. This course is for anyone wanting to gain proficiency in all aspects of parallel and distributed programming. Students develop skills necessary for understanding the design and analysis of parallel and distributed algorithms, and to appreciate not only the advantages, but also the difficulties of adapting algorithms to a parallel or distributed paradigm. The course is primarily a lecture-based course with the student required to complete significant projects outside of class time. The course includes individual as well as group projects and provide opportunities for the presentation and defense of designed solutions. The course encourages the development of professional communication skills and provides opportunities for collaborative project development.

Parallel computing is a critical component of the computing technology of the 21st century, and is likely to grow in importance with the proliferation of multiprocessor PC desktops and servers and scalable clusters of commodity workstations. This course examines the organizing principles behind parallel computing both from an architectural and a programming perspective. The course consists of two parts, organized around a common set of issues relevant to all parallel systems: naming, synchronization, latency, and bandwidth. The first part discusses how modern parallel computer architectures deal with these issues, both at the small (shared memory multiprocessors) and large (scalable multiprocessors) scales. The second part of the course discusses how the issues are dealt with in several common programming paradigms including message-passing, shared-memory, data-parallel, as well as higher-level approaches. The focus in this part of the course is on both programming techniques and programming for performance.

IV. Requirements Fulfilled

COSC 4311: Parallel Computing satisfies three hours of the requirements for the degree in computer science. It is required of all students pursuing a degree program in computer science within the College of Information Technology. This course may be used as an elective in the Information Technology and Computer engineering degree programs. This course should be taken in the senior year.

V. Required Prerequisites

COSC 3351: Algorithms

VI. Learning Outcomes

In this course, students learn:

- To develop an understanding of various basic concepts associated with parallel computing environments.
- To understand the effects that issues of synchronization, latency and bandwidth have on the efficiency and effectiveness of parallel computing applications.
- To gain experience in a number of different parallel computing paradigms including memory passing, memory sharing, data-parallel and other approaches.
- To earn experience in designing and testing parallel computing solutions to programming problems.
- To develop improved communication and collaborative skills.

VII. Assessment Strategy

The course grade involves an assessment of student performance on examinations that focus on the understanding of various concepts and constructs underlying Artificial Intelligence, and the communication of those concepts and the characteristics of designed solutions to AI problems to an audience. Course grades are based on:

- Weekly assigned homework to motivate students to do the work and earn credit accordingly.
- Weekly, in-class presentations by students related to independent literature research on aspects of the course material and classroom discussion and critique of the presentation.
- Two in-class examinations to assess the student's accumulative mastery of content covered prior to the time of the examination.
- Four programming assignments testing students understanding of the major concepts introduced during the course.
- A comprehensive final examination to assess the student's accumulative mastery of course material.

The final grade is based on 10% credit for the homework, 10% for the presentations and participation in classroom discussion, 20% on in-class examinations, 40% on programming assignments and 20% for the final examination.

Students are required to maintain a journal of thoughts and commentaries during the course. The journal contains daily entries including the identification of areas of interest and concern, notes on the preparation of presentation and comments and analysis of classmate's presentations. The journal is reviewed weekly by the instructor to provide feedback to the students.

Final grades and the student and instructor observations from reflective notebooks are included in the student's portfolio for use in the final assessment capstone course. The intent is to document the student's maturation as he proceeds through the curriculum.

VIII. Course Format

A. Instruction

This course is primarily a lecture/discussion course. Students are expected to attend three hours of lecture/discussion per week. At least once per week students should be prepared to make presentation summarizing an aspect of the AI literature selected by the instructor and to take part in a discussion based on that presentation. Once a week students should have at least 30 minutes of collaborative problem solving activity. Students should expect make significant use of the university's parallel computing facility outside of lecture class hours.

B. Web supplement

Course home page (the university's Web tool, WebCT or Blackboard) should contain the following:

- Course syllabus
- Course assignments
- Sample solutions to examinations (after being graded and returned)
- Sample solutions to programming assignments (after being graded and returned)
- Course calendar (an active utility)
- Course e-mail (an active utility)
- Course discussion list (an active utility)
- Student course performance (an active utility)

Classroom Hours (3 hours per week) **Class:** 3
Lab: 0

IX. Topics to Be Covered

A. Introduction, theoretical basis, system organization, and performance
B. Introduction to message passing
C. Programming with MPI
D. Mesh based computations, performance modeling
E. Performance measurement, scalability, projects
F. MPI datatypes; interconnection networks
G. Matrix multiplication; MPI communicators
H. High performance collective communication
I. Parallel programming languages , OpenMP, POSIX threads
J. Parallel sorting
K. Shared memory architecture and programming
L. Message passing implementation
M. Alternative programming models
N. Directions in high performance computer architecture

X. Laboratory Exercises

This course does not require a separate lab.

XI. Technology Component

This course makes use of the university's wireless access infrastructure. The course relies on the university and the students having access to professional grade application development environments for the students to use.

XII. Special Projects / Activities

Students are required to keep a "reflective notebook" in which, after each class, they enter their own assessments of what they learned, and what questions remain from the class. From each exercise set, each student selects one problem, which the student thinks best reflects the way the topic is used in a technical context. A detailed solution to the problem is included in the student's reflective notebook.

XIII. Textbooks and Teaching Aids

A. Required Textbook

Grama, A., Gupta, A., Karypis, G., and Kuman, V. *Introduction to Parallel Computing*, 2nd Edition, _____: Addison-Wesley, 2003.
ISBN:

B. Alternative Textbooks

None.

C. Supplemental Print Materials

1. Wilkinson, B., and M. Allen. *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers*. _____: Prentice-Hall, 2000.
   ISBN:
2. Pacheco, Peter. *Parallel Programming with MPI*. _____: Morgan-Kaufmann, 1996.
   ISBN:

D. Supplemental Online Materials

None.

Course Title: COSC 4361: Operating Systems

Semester Credit Hours: 3 (3,0)

I. Course Overview

This course is the study of the principles, purposes, and organization of operating systems. The goal is to prepare students an understanding of the theory as well as practices of the design and implementation of operating systems software.

II. Competencies Addressed

Students in this course develop conceptual and programming skills necessary for continued success in computer science. The skills enhances their abilities to appreciate the theory and practices of operating systems common to computer science as a discipline and to effectively communicate their solutions to fellow professionals. This course makes extensive use of the PMU technology infrastructure to provide communication between faculty and students. The course includes individual as well as group projects, establishes both conceptual reasoning skills and technical communication skills, and provides opportunities for the presentation and defense of designed solutions.

III. Detailed Course Description

COSC 4361: Operating Systems is concerned with the study of the principles, purposes, and organization of operating systems, including processes, tasks, scheduling, interprocess communication, synchronization, mutual exclusion, memory management, device management, file systems, security and protection, multi-CPU systems, computer networking, and distributed computing.

IV. Requirements Fulfilled

This course satisfies three hours of the requirements for the degree in computer science. It is required of all students pursuing a degree program in computer science within the College of Information Technology. It should be taken in the first semester of the senior year.

V. Required Prerequisites

- GEIT 1311: Computer Organization
- COSC 3411: System Programming
- COSC 3421: Data Structures

VI. Learning Outcomes

In this course, students learn:

- To understand the underlying system structures of operating systems.
- To understand the concepts of threads and processes in computer systems.
- To learn the design requirements on storage management in computer systems.
- To develop improved communication and collaborative skills.

VII. Assessment Strategy

This course is designed with three primary goals in mind: to introduce students to the conceptual basis and practical issues associated with the compiler construction, to enhance the student's programming techniques to its application in computer science, and to provide students with the opportunity to communicate their designs and implementations to their peers in a professional setting. With this in mind, the course grade involves an assessment of their performance on examinations that focus on the application of programming paradigms to the solutions of problems, the performance analysis of the designed solutions, and the communication of designed solutions to those problems to an audience. Course grades are based on:

- Weekly assigned homework to motivate students to do the work and earn credit accordingly.
- Weekly, in-class presentations by students of solutions to real world problems related to the course material and classroom discussion and critique of the presentation.
- Two in-class examinations to assess the student's accumulative mastery of content covered prior to the time of the examination.
- Three programming assignments testing students understanding of the major concepts introduced during the course.
- A comprehensive final examination to assess the student's accumulative mastery of course material.

The final grade is based on 10% credit for the homework, 10% for the presentations and participation in classroom discussion, 30% on in-class examinations, 30% on programming assignments and 20% for the final examination.

Students are required to maintain a journal of thoughts and commentaries during the course. The journal contains daily entries including the identification of areas of interest and concern, notes on the preparation of presentation and comments and analysis of classmate's presentations. The journal is reviewed weekly by the instructor to provide feedback to the students.

Final grades and the student and instructor observations from reflective notebooks are included in the student's portfolio for use in the final assessment capstone course. The intent is to document the student's maturation as he proceeds through the curriculum.

VIII. Course Format

A. Instruction

This course utilizes both lecture/discussion and laboratory exercises. Students are expected to attend three hours of lecture per week. At least once per week students should be prepared to make presentation on the design and implementation of a solution to a problem selected by the instructor and to take part in a discussion based on that presentation. Once a week students should have at least 30 minutes of collaborative problem solving activity.

B. Web supplement

Course home page (the university's Web tool, WebCT or Blackboard) should contain the following:

- Course syllabus
- Course assignments
- Sample solutions to examinations (after being graded and returned)
- Sample solutions to programming assignments (after being graded and returned)
- Course calendar (an active utility)
- Course e-mail (an active utility)
- Course discussion list (an active utility)
- Student course performance (an active utility)

Classroom Hours (3 hours per week) **Class:** 3
Lab: 0

IX. Topics to Be Covered

A. Basic Concepts
   1. Computer system structures
   2. Operating system structures

B. Threads and processes
   1. Processes, threads, and address spaces
   2. Thread and process coordination
   3. CPU scheduling
   4. Deadlocks

C. Storage management
   1. Memory management
   2. Virtual memory
   3. File systems
   4. Input-output systems
   5. Mass storage structure

X. Laboratory Exercises

This course does not offer a separate laboratory to students.

XI. Technology Component

This course makes use of the university's wireless access infrastructure. The course relies on the university and the students having access to professional grade application development environments for the students to use.

XII. Special Projects / Activities

Students are required to keep a "reflective notebook" in which, after each class, they enter their own assessments of what they learned, and what questions remain from the class. From each exercise set, each student selects one problem, which the student thinks best reflects the way the topic is used in a technical context. A detailed solution to the problem is included in the student's reflective notebook.

XIII. Textbooks and Teaching Aids

A. Required Textbook

Silberschatz, A., P., B. Galvin, G. Gagne. *Operating Systems Concepts*. _____: Wiley, 2002.
ISBN: 0-471-41743-2

B. Alternative Textbooks

Tanenbaum, A. *Modern Operating Systems*. _____: Prentice Hall, 2001.
ISBN 0-13-031358-0.

C. Supplemental Print Materials

None

D. Supplemental Online Materials

As available from the publishers.

Course Title: COSC 4362: Artificial Intelligence

Semester Credit Hours: 3 (3,0)

I. Course Overview

The course presents an overview of artificial intelligence and its methods for solving problems. Basic algorithms for finding solutions to problems or adaptively improving responses to situations are discussed. Expert systems, genetic algorithms, and intelligent agents are among the areas that are explored.

II. PMU Competencies and Learning Outcomes

Students of COSC 4362: Artificial Intelligence develop skills necessary for understanding the design of artificial intelligence applications so as to appreciate the strengths and limitations of artificial intelligence approaches to problem solving. This course makes extensive use of the PMU technology infrastructure to provide communication between faculty and students. The course is primarily a lecture-based course with the student required to complete significant projects outside of class time. The course includes individual as well as group projects and provides opportunities for the presentation and defense of designed solutions. The course encourages the development of professional communication skills and provides opportunities for collaborative project development.

III. Detailed Course Description

This course examines artificial intelligence as a tool for solving problems. The course looks at the major categories (application domains) of artificial intelligence (AI), including vision, natural language, planning and others with a view to identifying characteristics that mark domains as potential areas of interest. An overview of search algorithms is presented with an emphasis on heuristic driven search, including the A* algorithm and game trees. The course discusses approaches to knowledge representation, including those based on predicate calculus and non-symbolic representations like neural networks. It introduces the concepts of expert systems, intelligent agents and genetic algorithms.

IV. Requirements Fulfilled

COSC 4362: Artificial Intelligence satisfies three hours of the requirements for the degree in computer science. It is required of all students pursuing a degree program in computer science within the College of Information Technology. This course may be used as an elective in the Information Technology and Computer engineering degree programs. This course should be taken in the senior year.

V. Required Prerequisites

- COSC 3421: Data Structures
- COSC 3351: Algorithms

VI. Learning Outcomes

In this course, students learn:

- To develop an understanding of various basic concepts and constructs underlying the artificial intelligence.
- To earn experience in designing and testing AI solutions to programming problems.
- To become familiar with a variety of common AI developmental tools
- To be able to discuss the strengths and limitations of different AI paradigms in solving programming problems.
- To develop improved communication and collaborative skills.

VII. Assessment Strategy

The course grade involves an assessment of student performance on examinations that focus on the understanding of various concepts and constructs underlying Artificial Intelligence, and the communication of those concepts and the characteristics of designed solutions to AI problems to an audience. Course grades are based on:

- Weekly assigned homework to motivate students to do the work and earn credit accordingly.
- Weekly, in-class presentations by students related to independent literature research on aspects of the course material and classroom discussion and critique of the presentation.
- Four written reports summarizing literature related to aspects of the course.
- Two in-class examinations to assess the student's accumulative mastery of content covered prior to the time of the examination.
- Three programming assignments testing students understanding of the major concepts introduced during the course.
- A comprehensive final examination to assess the student's accumulative mastery of course material.

The final grade is based on 10% credit for the homework, 10% for the presentations and participation in classroom discussion, 20% for written reports, 20% on in-class examinations, 30% on programming assignments and 10% for the final examination.

Students are required to maintain a journal of thoughts and commentaries during the course. The journal contains daily entries including the identification of areas of interest and concern, notes on the preparation of presentation and comments and analysis of classmate's presentations. The journal is reviewed weekly by the instructor to provide feedback to the students.

Final grades and the student and instructor observations from reflective notebooks are included in the student's portfolio for use in the final assessment capstone course. The intent is to document the student's maturation as he proceeds through the curriculum.

VIII. Course Format

A. Instruction

This course is primarily a lecture/discussion course. Students are expected to attend three hours of lecture/discussion per week. At least once per week students should be prepared to make presentation summarizing an aspect of the AI literature selected by the instructor and to take part in a discussion based on that presentation. Once a week students should have at least 30 minutes of collaborative problem solving activity. Students should expect to incorporate a significant independent experimentation with AI development tools.

B. Web supplement

Course home page (the university's Web tool, WebCT or Blackboard) should contain the following:

- Course syllabus
- Course assignments
- Sample solutions to examinations (after being graded and returned)
- Sample solutions to programming assignments (after being graded and returned)
- Course calendar (an active utility)
- Course e-mail (an active utility)
- Course discussion list (an active utility)
- Student course performance (an active utility)

Classroom Hours (3 hours per week) **Class:** 3
Lab: 0

IX. Topics to Be Covered

A. Overview
   1. Definition of terms
   2. Major divisions of AI
B. Searching for solutions
   1. Data tree search algorithms
   2. Heuristics
   3. A* algorithms
   4. Game trees
C. Knowledge representations
   1. Symbolic vs. non-symbolic predicate calculus
   2. Forward and backward chaining

D. Systems
    1. Expert systems
    2. Genetic classifier systems
E. Machine learning
    1. Non-symbolic representations
    2. Genetic algorithms
    3. Neural networks
    4. Reinforcement learning methods
F. Planning vs. reacting
    1. Intelligent agents
    2. Robotics

X. Laboratory Exercises

This course does not require a separate laboratory.

XI. Technology Component

This course makes use of the university's wireless access infrastructure. The course relies on the university and the students having access to professional grade application development environments for the students to use. The course has a laboratory component that would be best implemented in university provided laboratory space.

XII. Special Projects / Activities

Students are required to keep a "reflective notebook" in which, after each class, they enter their own assessments of what they learned, and what questions remain from the class. From each exercise set, each student selects one problem, which the student thinks best reflects the way the topic is used in a technical context. A detailed solution to the problem is included in the student's reflective notebook.

XIII. Textbooks and Teaching Aids

A. Required Textbook

Luger, G.F. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. _____: Pearson Addison Wesley, 2002. ISBN: 0-201-64866-0.

B. Alternative Textbooks

Norvig, P. *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp*. _____: Morgan Kaufmann, 1991. ISBN: 1-558-60191-0

C. Supplemental Print Materials

None.

D. Supplemental Online Materials

1. PDProlog Interpreter: http://www-2.cs.cmu.edu/afs/cs/project/ai-repository/ai/lang/prolog/impl/prolog/pdprolog/0.html
2. Xlisp: http://www.aracnet.com/~tomalmy/xlisp.html
3. CLIPS (public domain expert system): http://www.ghgcorp.com/clips/OtherWeb.html

Course Title: COSC 4363: Automata Theory

Semester Credit Hours: 3 (3,0)

I. Course Overview

This course is to give an introductory study of automata, formal languages, and computability, including set theory and countability, finite automata and regular languages, push-down automata and context-free languages, Turing machines, Church's thesis, halting problem, and uncomputability.

II. PMU Competencies and Learning Outcomes

Students in this course develop quantitative skills necessary for continued success in computer science. The skills enhance their abilities to analyze and comprehend mathematically the design specifications of programming problems common to computer science as a discipline and to effectively communicate their solutions to fellow professionals. This course makes extensive use of the PMU technology infrastructure to provide communication between faculty and students. The course includes individual as well as group projects, establishes both mathematical reasoning skills and technical communication skills, and provides opportunities for the presentation and defense of designed solutions.

III. Detailed Course Description

COSC 4363 is concerned with the an introductory study of automata, formal languages, and computability, including set theory and countability, finite automata and regular languages, push-down automata and context-free languages, Turing machines, Church's thesis, halting problem, and uncomputability. One important lasting effect of this course is to develop the ability and reasoning to understand mathematically the design specifications of programming problems presented in this course.

IV. Requirements Fulfilled

COSC 4363: Automata Theory satisfies three hours of the requirements for the degree in computer science. It is required of all students pursuing a degree program in computer science within the College of Information Technology. It should be taken in the second semester of the junior year.

V. Required Prerequisites

- COSC 3351: Algorithms
- MATH 2331: Linear Algebra
- MATH 2332: Differential Equations

VI. Learning Outcomes

In this course, students learn:

- To understand set theory and countability.
- To understand and apply finite automata and regular languages.
- To understand and apply push-down automata and context-free languages.
- To understand and appreciate the issues of computability.
- To develop improved communication and collaborative skills.

VII. Assessment Strategy

This course is designed with three primary goals in mind: to further students the understanding of automata theory, to lead students to connect the mathematics to its application in computer science, and to provide students with the opportunity to communicate their ideas and their expertise to the professional community. With this in mind, the course grade involves an assessment on in-class quizzes and examinations that focus on the applications of automata theory to computer science. Course grades are based on:

- Weekly assigned homework to motivate students to do the work and earn credit accordingly.
- Weekly in-class quizzes.
- Weekly, in-class presentations by students of solutions to real world problems related to the course material and classroom discussion and critique of the presentation.
- Two in-class examinations to assess the student's accumulative mastery of content covered prior to the time of the examination.
- A comprehensive final examination to assess the student's accumulative mastery of course material.

The final grade is based on 15% credit for the homework, 15% for the quizzes, 10% for the presentations and participation in classroom discussion, 30% on in-class examinations, and 30% for the final examination.

Students are required to maintain a journal of thoughts and commentaries during the course. The journal contains daily entries including the identification of areas of interest and concern, notes on the preparation of presentation and comments and analysis of classmate's presentations. The journal is reviewed weekly by the instructor to provide feedback to the students.

Final grades and the student and instructor observations from reflective notebooks are included in the student's portfolio for use in the final assessment capstone course. The intent is to document the student's maturation as he proceeds through the curriculum.

VIII. Course Format

A. Instruction

This course utilizes lecture as the primary instruction. Students are expected to attend three hours of lecture per week. At least once per week students should be prepared to make presentation on the design and implementation of a solution to a problem selected by the instructor and to take part in a discussion based on that presentation. Once a week students should have at least 30 minutes of collaborative problem solving activity.

B. Web supplement

Course home page (the university's Web tool, WebCT or BLACKBOARD) should contain the following:

- Course syllabus.
- Course assignments.
- Sample solutions to examinations (after being graded and returned).
- Sample solutions to programming assignments (after being graded and returned).
- Course calendar (an active utility).
- Course e-mail (an active utility).
- Course discussion list (an active utility).
- Student course performance (an active utility).

Classroom Hours (3 hours per week) **Class: 3**
Lab: 0

IX. Topics to be Covered

A. Set, relations, and functions
B. Regular languages and finite automata
   a. Deterministic finite automata (DFAs)
   b. Non-deterministic finite automata (NFAs)
   c. Regular languages
   d. Closure properties
   e. Pumping lemmas
C. Context-free languages (CFLs) and pushdown automata (PDAs)
   a. CFLs, PDAs, and their equivalence
   b. Regular grammars
   c. Closure properties
   d. Pumping lemmas
D. Turing machines
   a. Definitions and examples
   b. Turing decidability, computability, and acceptability
   c. Combining Turing machines
   d. Turing machine variants
   e. Universal Turing machines
   f. Church's Thesis
E. Undecidability

X. Laboratory Exercises

This course does not offer a separate laboratory to students.

XI. Technology Component

This course makes use of the university's wireless access infrastructure. The course relies on the university and the students having access to professional grade application development environments for the students to use.

XII. Special Projects/Activities

Students are required to keep a "reflective notebook" in which, after each class, they enter their own assessments of what they learned, and what questions remain from the class. From each exercise set, each student selects one problem, which the student thinks best reflects the way the topic is used in a technical context. A detailed solution to the problem is included in the student's reflective notebook.

XIII. Textbooks and Teaching Aids

A. Required Textbook

H. Lewis and C. Papadimitriou, *Elements of the Theory of Computation*, (1997) Prentice Hall
ISBN 0-12-03293-7

B. Alternative Textbooks

J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation* (2001), Addison-Wesley
ISBN 0-201-44124-1

C. Supplemental Print Materials

None

D. Supplemental Online Materials

As available from publishers.

Course Title: COSC 4364: Compilers

Semester Credit Hours: 3 (3,0)

I. Course Overview

This course is the study of the theory and practice of constructing a compiler, including lexical analysis, parsing, semantic analysis, run-time organization, code generation, and optimization. During the course of the semester, the students complete a significant compiler project.

II. PMU Competencies and Learning Outcomes

Students in this course develop conceptual and programming skills necessary for continued success in computer science. The skills enhance their abilities to appreciate the theory and practices of compiler construction common to computer science as a discipline and to effectively communicate their solutions to fellow professionals. This course makes extensive use of the PMU technology infrastructure to provide communication between faculty and students. The course includes individual as well as group projects, establishes both conceptual reasoning skills and technical communication skills, and provides opportunities for the presentation and defense of designed solutions.

III. Detailed Course Description

COSC 4364 is concerned with the study of the theory and practice of constructing a compiler, including lexical analysis, parsing, semantic analysis, run-time organization, code generation, and optimization. During the course of the semester, the students complete a significant compiler project. The techniques studied in this course are particularly useful for programming applications that are the input of other programs. Thus, the course enriches students to formulate, design, and implement solutions to open programming problems.

IV. Requirements Fulfilled

COSC 4364: Compilers satisfies three hours of the requirements for the degree in computer science. It is required of all students pursuing a degree program in computer science within the College of Information Technology. Students are recommended to take the course in the second semester of the senior year.

V. Required Prerequisites

- COSC 3351: Algorithms
- COSC 4461: Programming Languages

VI. Learning Outcomes

In this course, students learn:

- To understand and apply the phases in compilation process.
- To understand the concept of lexical analysis, and apply the tools for lexical analysis.
- To acquire the concept of parsing, and apply the tools for parsing.
- To learn and apply the concept of semantic analysis.
- To understand the run-time environments in compilation.
- To acquire the knowledge of code generation and optimization.
- To develop improved communication and collaborative skills.

VII. Assessment Strategy

This course is designed with three primary goals in mind: to introduce students to the conceptual basis and practical issues associated with the compiler construction, to enhance students' programming techniques to its application in computer science, and to provide students with the opportunity to communicate their designs and implementations to their peers in a professional setting. With this in mind, the course grade involves an assessment of their performance on examinations that focus on the application of programming paradigms to the solutions of problems, the performance analysis of the designed solutions, and the communication of designed solutions to those problems to an audience. Course grades are based on:

- Weekly assigned homework to motivate students to do the work and earn credit accordingly.
- Weekly, in-class presentations by students of solutions to real world problems related to the course material and classroom discussion and critique of the presentation.
- One in-class examination to assess the student's accumulative mastery of content covered prior to the time of the examination.
- 5 programming assignments testing students understanding of the major concepts introduced during the course.
- A comprehensive final examination to assess the student's accumulative mastery of course material.

The final grade is based on 10% credit for the homework, 10% for the presentations and participation in classroom discussion, 10% on in-class examinations, 50% on programming assignments and 20% for the final examination.

Students are required to maintain a journal of thoughts and commentaries during the course. The journal contains daily entries including the identification of areas of interest and concern, notes on the preparation of presentation and comments and analysis of classmate's presentations. The journal is reviewed weekly by the instructor to provide feedback to the students.

Final grades and the student and instructor observations from reflective notebooks are included in the student's portfolio for use in the final assessment capstone course. The intent is to document the student's maturation as he or she proceeds through the curriculum.

VIII. Course Format

This course utilizes both lecture/discussion and laboratory exercises. Students are expected to attend three hours of lecture per week. At least once per week students should be prepared to make presentation on the design and implementation of a solution to a problem selected by the instructor and to take part in a discussion based on that presentation. Once a week students should have at least 30 minutes of collaborative problem solving activity.

Classroom Hours (3 hours per week)	**Class: 3** **Lab: 0**

Web supplement:

Course home page (the university's Web tool, WebCT or Blackboard) should contain the following:

- Course syllabus
- Course assignments
- Sample solutions to examinations (after being graded and returned)
- Sample solutions to programming assignments (after being graded and returned)
- Course calendar (an active utility)
- Course e-mail (an active utility)
- Course discussion list (an active utility)
- Student course performance (an active utility)

IX. Topics to be Covered

A. Phases in compilation process
B. Lexical analysis
    a. Regular expressions and finite state automata (FSAs)
    b. Non-deterministic finite state automata (NFAs)
    c. Subset construction
    d. Thompson's construction
C. Parsing
    a. Bottom-up parsing
    b. LR parsing algorithm
    c. Semantic actions
    d. Abstract syntax trees
    e. SLR parser
    f. Top-down parsing

D. Semantic analysis
   a. Scopes
   b. Symbol tables
   c. Type-checking
E. Run-time organization
   a. Scoping and storage allocation
   b. Activation records
   c. Non-Local references
   d. Parameter-passing techniques
F. Code generation
   a. Intermediate code generation
   b. Final code generation
G. Machine-independent optimizations

X. Laboratory Exercises

This course does not offer a separate laboratory to students.

XI. Technology Component

This course makes use of the university's wireless access infrastructure. The course relies on the university and the students having access to professional grade application development environments for the students to use.

XII. Special Projects/Activities

Students are required to keep a "reflective notebook" in which, after each class, they enter their own assessments of what they learned, and what questions remain from the class. From each exercise set, each student selected one problem, which the student thinks best reflects the way the topic is used in a technical context. A detailed solution to the problem is included in the student's reflective notebook.

XIII. Textbooks and Teaching Aids

A. Required Textbook

T. W. Parsons, *Introduction to Compiler Construction*, (1992) W. H. Freeman
ISBN 0-7167-8261-8

B. Alternative Textbooks

None

C. Supplemental Print Materials

None

D. Supplemental Online Materials

As available from publisher.

Course Title: COSC 4461: Programming Languages

Semester Credit Hours: 4 (3,1)

I. Course Overview

Programming languages is the study of basic concepts and constructs underlying the design of the modern programming languages. Various programming paradigms, including object-oriented, functional, logic, and concurrent programming, are discussed.

II. PMU Competencies and Learning Outcomes

Students in this course develop skills necessary for understanding the design of the modern programming languages so as to appreciate the strengths and limitations among different programming paradigms. These skills are necessary for continued success in computer science. This course makes extensive use of the PMU technology infrastructure to provide communication between faculty and students. The course includes a structured laboratory component to ensure that students gain the necessary experience and skill in managing the concepts introduced in the class. The course includes individual as well as group projects and provide opportunities for the presentation and defense of designed solutions.

III. Detailed Course Description

COSC 4461: Programming Languages is concerned with the study of basic concepts, including types, control structures, abstraction mechanisms, inheritance, concurrency, and constructs underlying the design of the modern programming languages. Various programming paradigms, including object-oriented, functional, logic, and concurrent programming, are discussed. Students are exposed to the implementation of programs using programming languages under different programming paradigms. One important lasting effect of this course is to enhance and develop the ability to design, implement and test solutions to effectively solve programming problems utilizing appropriate programming languages.

IV. Requirements Fulfilled

This course satisfies four hours of the requirements for the degree in computer science. It is required of all students pursuing a degree program in computer science within the College of Information Technology. It should be taken in the first semester of the junior year.

V. Required Prerequisites

- GEIT 1412: Computer Science II
- COSC 3411: System Programming

VI. Learning Outcomes

In this course, students learn:

- To develop an understanding of various basic concepts and constructs underlying the design of the modern programming languages.
- To earn experience in designing and testing solutions to programming problems implemented with various programming paradigms.
- To be able to discuss the strengths and limitations of different programming paradigms in solving programming problems.
- To develop improved communication and collaborative skills.

VII. Assessment Strategy

This course is designed with three primary goals in mind: to introduce students to the conceptual basis and practical issues associated with the use of various programming paradigms to solve programming problems, to appreciate the strengths and limitations of different paradigms, and to provide students with the opportunity to communicate their designs and implementations to their peers in a professional setting. With this in mind, the course grade involves an assessment of their performance on examinations that focus on the understanding of various concepts and constructs underlying the design of the modern programming languages, and the communication of designed solutions to those problems to an audience. Course grades are based on:

- Weekly assigned homework to motivate students to do the work and earn credit accordingly.
- Weekly, in-class presentations by students of solutions to real world problems related to the course material and classroom discussion and critique of the presentation.
- Weekly structured laboratory exercises designed to guide students through specific course topics.
- Two in-class examinations to assess the student's accumulative mastery of content covered prior to the time of the examination.
- 5 programming assignments testing students understanding of the major concepts introduced during the course.
- A comprehensive final examination to assess the student's accumulative mastery of course material.

The final grade is based on 10% credit for the homework, 10% for the presentations and participation in classroom discussion, 20% for weekly laboratory exercises, 20% on in-class examinations, 30% on programming assignments and 10% for the final examination.

Students are required to maintain a journal of thoughts and commentaries during the course. The journal contains daily entries including the identification of areas of interest and concern, notes on the preparation of presentation and comments and analysis of classmate's presentations. The journal is reviewed weekly by the instructor to provide feedback to the students.

Final grades and the student and instructor observations from reflective notebooks are included in the student's portfolio for use in the final assessment capstone course. The intent is to document the student's maturation as he proceeds through the curriculum.

VIII. Course Format

A. Instruction

This course utilizes both lecture/discussion and laboratory exercises. Students are expected to attend three hours of lecture/discussion per week and two hours of laboratory per week. At least once per week students should be prepared to make presentation on the design and implementation of a solution to a problem selected by the instructor and to take part in a discussion based on that presentation. Once a week students should have at least 30 minutes of collaborative problem solving activity.

B. Web supplement

Course home page (the university's Web tool, WebCT or Blackboard) should contain the following:

- Course syllabus
- Course assignments
- Sample solutions to examinations (after being graded and returned)
- Sample solutions to programming assignments (after being graded and returned)
- Course calendar (an active utility)
- Course e-mail (an active utility)
- Course discussion list (an active utility)
- Student course performance (an active utility)

Classroom Hours (5 hours per week) **Class:** 3
Lab: 2

IX. Topics to be Covered

A. Variety of programming languages
   A. Abstraction
   B. Compilers and interpreters
   C. Syntax and semantics
   D. Context-free grammars
B. Imperative languages
   A. Control structures
   B. Data types and their presentation
   C. Composite types
   D. Subprograms, functions, procedures, methods
   E. Program structure
   F. Exception handling
C. Programming paradigms
   A. Object-oriented programming
   B. Functional programming
   C. Logic programming
   D. Concurrent programming

X. Laboratory Exercises

This course requires a weekly 2-hour laboratory component. Topics to be covered in the laboratory sessions should include:

- Compilers – exercises in using various programming language compilers in UNIX.
- make Utility – exercises in using the UNIX make utility.
- LISP I – exercises in familiarizing the use of the functional programming language LISP.
- LISP II – additional exercises in familiarizing the use of LISP.
- Prolog I – exercises in familiarizing the use of the logic programming language Prolog.
- Prolog II – additional exercises in familiarizing the use of Prolog.
- Java I – exercises in familiarizing the use of Java.
- Java II – additional exercises in familiarizing the use of Java.
- Constructors and Destructors – exercises in using the constructors and destructors in C++.
- List Manipulation – exercises in performing list manipulations in LISP.
- Unification – exercises in performing unifications in Prolog.
- Concurrency – exercises in exploiting concurrency in Java.
- Three additional lab sessions should be kept in reserve to allow the instructor to extend the more difficult laboratories for more than one session.

XI. Technology Component

This course makes use of the university's wireless access infrastructure. The course relies on the university and the students having access to professional grade application development environments for the students to use. The course has a laboratory component that would be best implemented in university provided laboratory space.

XII. Special Projects/Activities

Students are required to keep a "reflective notebook" in which, after each class, they enter their own assessments of what they learned, and what questions remain from the class. From each exercise set, each student selects one problem, which the student thinks best reflects the way the topic is used in a technical context. A detailed solution to the problem is included in the student's reflective notebook.

XIII. Textbooks and Teaching Aids

A. Required Textbook

Sethi, R.. *Programming Languages: Concepts and Constructs*. _____: Addison-Wesley,1995.
ISBN 0-201-59065-4

B. Alternative Textbooks

Krishnamurthi, S. *Programming Languages: Application and Interpretation*. _____: _____, 2003.
Available at:
http://www.cs.brown.edu/~sk/Publications/Books/ProgLangs

C. Supplemental Print Materials

None.

D. Supplemental Online Materials

As available from the publisher.