Computer Science Master Syllabi (2016)

1. Course: CSCI 1301 Computer Science I

2. Credit Hours: 3 hrs Contact Hours: 150 minutes each week (50/50/50 or 75/75)

3. Course Coordinator: Dr. Gita Phelps

4. Text: Starting out with Java from Control Structures through Data Structures, Pearson Education Inc (Addison Wesley), by Tony Gaddis and Godfrey Muganda Third edition (2016)

   a. CodeLab: http://www.turingscraft.com/ You are required to purchase access to CodeLab via the website as a part of this class.

5. Specific course information

   a. Course topics include an overview of computers and programming; problem-solving and algorithm development; simple data types; arithmetic and logical operators; selection structures; text files; arrays; procedural abstraction and software design; modular programming.

   b. Prerequisites or Co-requisites: None

   c. Required

6. Specific goals for the course

   a) Outcomes of instruction

      1. Be able to use a high level programming language to implement solutions to a variety of elementary algorithmic solutions to problems
      2. Become acquainted with the basics of computers and programming
      3. Be able to write programs using the basics of control structures
      4. Be able to understand and apply concepts of modular programming using methods
      5. Be able to understand and apply concepts of object oriented programming
      6. Be able to use other data types, such as classes, arrays and files.

   b) . Student Outcome

      SO 2: An ability to apply design and development principles in the construction of software systems.

   c) ABET Enabled Outcomes

      ABET (b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution

7. Brief list of topics to include but not limited to.

   * An overview of computers and programming;
   * Problem solving and algorithm development;
   * Modular programming;
   * Simple data types;
   * Composite data types (classes);
   * Arithmetic and logical operators;
   * Sequential structures;
   * Selection structures;
   * Repetition structures;

1. Course: CSCI 1302 Computer Science II

2. Credit Hours: 3 hrs Contact Hours: 150 minutes each week (50/50/50 or 75/75)

3. Course Coordinator: Dr. Gita Phelps

4. Text: Starting out with Java from Control Structures through Data Structures, Pearson Education Inc (Addison Wesley), by Tony Gaddis and Godfrey Muganda Third edition (2016)
   a. Phidgets: http://www.phidgets.com

5. Specific course information
   a. The course includes an overview of abstract data types; multi-dimensional arrays and records; strings; binary files; searching and sorting; software engineering concepts; software engineering concepts; dynamic data structures; introduction to object oriented languages and the concepts of object oriented design of algorithms.
   b. Prerequisites: C or better in CSCI 1301
   c. Required

6. Specific goals for the course
   a) Outcomes of instruction
      1. Compare and contrast static class members and instance class members.
      2. Explain object referencing when comparing and copying objects and passing/returning objects to/from methods.
      3. Construct modular, well-structured programs that reuse classes through aggregation and inheritance.
      4. Demonstrate an understanding of overloading and overriding methods by using them appropriately in a program
      5. Split a string into tokens using the split method
      6. Explain the use of abstract classes and interfaces.
      7. Write a program that gracefully responds to exceptions when they are thrown with exception handling.
      8. Compare and contrast text files and binary files and describe differences in program code implementation.
      9. Create a program with a graphical user interface with various GUI components.
      10. Create both java applets and applications.
   b) Student Outcomes
      SO 2: An ability to apply design and development principles in the construction of software systems.
      SO 5: An ability to communicate effectively with a range of audiences.
   c) ABET Enabled Outcomes
      ABET (b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution
      ABET (f) An ability to communicate effectively with a range of audiences

7. Brief list of topics to include but not limited to:
   * Abstract data types
   * Advanced class design
   * Strings and Wrapper class
   * File I/O
   * Exceptions
   * GUI applications
   * Concepts of object oriented programming, including encapsulation, inheritance, and polymorphism

1. Course: **CSCI 2350 Objected Oriented Programming**
2. Credit Hours: 3 150 minutes per week, 16 weeks per semester
3. Course Coordinator: Mr. Patrick Simmons
4. Text: : Starting Out with C++: From Control Structures through Objects (7th Edition) by Tony Gaddis
   SUPPLEMENT MATERIAL: * Online resources about C++
5. Specific course Information
   a) Students are introduced to C++. In addition to learning the aspects of procedural programming languages, students will be strengthened in the knowledge of object oriented languages and the concepts of object oriented design of algorithms. The central issues of data hiding, encapsulation, inheritance, and polymorphism are emphasized.
   b) C or better in CSCI 1302 (Computer Science II)
   c) Required:

6. Specific goals for the course
   a) Outcomes of instruction
      1. Students will become familiar with the programming language, C++;
      2. Students will understand the procedural aspects of the programming language, such as
         - o Expressions and Interactivity
         - o Functions
         - o Array
         - o Searching
         - o Sorting
         - o Variables
         - o Data types (including generic)
         - o Loops
         - o File Operations
         - o Recursion
         - o Pointer and Linked Lists
      3. Students will understand the object oriented aspect of the programming language, such as:
         - o classes
         - o data encapsulation and data hiding
         - o composition, inheritance and software reusability
         - o create and manipulate super-classes and subclasses
         - o create use and destroy objects
         - o control access to object instance variables and methods
         - o appreciate how polymorphism makes systems extensible and maintainable
      4. Students will understand exceptions and error handling
      5. Students will be introduced to Standard Template Library (STL)
   b) Student Outcome
   None
   c) ABET Enabled Outcomes
   ABET (b): An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution
7. Brief list of topics to include but not limited to::

* Introduction to Computers & Programming
* Introduction to C++
* Expression and Interactivity
* Making Decisions
* Loops and Files
* Functions
* Recursion
* Arrays
* Searching and Sorting Arrays
* Characters, c-strings, string Class
* Structured Data
* Advanced File Operations
* Classes
* Inheritance, Polymorphism, and Virtual Functions
* Exceptions,
* Templates and Generic Data Types
* The Standard Template Library (STL)
* Pointers
* Lambdas
* Linked Lists

1. Course: CSCI 2800 Social and Professional Issues
2. Credit Hours: 3 hrs: 165 minutes each week
3. Course Coordinator: Mr. Daniel Gale
4. Text: A Gift of Fire - Social, Legal, and Ethical Issues for Computing Technology (4th edition)
By Sara Base PUBLISHER: Pearson ISBN: 9780132492676
5. Specific course information
   a. An introduction to software applications, equipment, information representation, information security, ethical issues, and the use of Internet information resources..
   b. Prerequisites: C or better in CSCI 1302
   c. Required

6. Specific goals for the course
   a) Outcomes of instruction
   1. Each student will have an understanding of professional issues and responsibilities.
   2. Each student will have an understanding of ethical issues and responsibilities.
   3. Each student will have an understanding of legal issues and responsibilities.
   4. Each student will have an understanding of security issues and responsibilities.
   5. Each student will have an understanding social issues and responsibilities.
   6. Each student will have an ability to analyze the local and global impact of computing on individuals, organizations, and society.

   b) Student Outcomes
      SO 4: An understanding of professional, ethical, legal, security and social issues and responsibilities.

   c) ABET Enabled Outcomes
      ABET (e) An understanding of professional, ethical, legal, security and social issues and responsibilities
      ABET (g) An ability to analyze the local and global impact of computing on individuals, organizations, and society

7. Brief list of topics to include but not limited to:
   - Privacy
   - Freedom of Speech
   - Intellectual Property
   - Crime
   - Work
   - Evaluating and Controlling Technology
   - Errors, Failures and Risks
   - Professional Ethics and Responsibilities

1. Course: CSCI 3211 Assembly Language & Digital Logic Design
2. Credit Hours: 3 hrs (Contact Hours: 75/75 150 Minutes)
3. Course Coordinator: Gongbing Hong
4. Text: Essentials of 80 x 86 Assembly Language by Richard Detmer 2nd Edition
5. Specific course information
   a. The internal organization of digital computers is introduced through the study of assembly language. Digital logic design, combinational and sequential circuit design and implementation are introduced. Laboratory projects include the implementation of circuits using integrated circuit technology and assembly language programming.
   b. Prerequisites: CSCI 1302 Co-requisites: CSCI 3410
   c. Required
6. Goals for course:
   a) Outcomes of instruction: Students will be able to :
      1. Use CAD tools for capture, synthesis, and simulation to evaluate simple building blocks (e.g., arithmetic logic unit, registers, movement between registers) of a simple computer design.
      2. Describe internal formats to represent numerical and non-numerical data.
      3. Write simple programs at the assembly/machine level for string processing and manipulation.
      4. Explain the organization of the classical von Neumann machine and its major functional units.
      5. Summarize how instructions are represented at both the machine level and in the context of a symbolic assembler.
      6. Demonstrate how to map between high-level language patterns into assembly/machine language notations.
      7. Explain different instruction formats, such as addresses per instruction and variable length vs. fixed length formats.
      8. Explain how subroutine calls are handled at the assembly level.
      9. Write simple assembly language program segments.
      10. Show how fundamental high-level programming constructs are implemented at the machine-language level.
   b) Student Outcomes
      SO 2: An ability to apply design and development principles in the construction of software systems.
   c) ABET Enabled Outcomes:
      ABET(k):An ability to apply design and development principles in the construction of software systems.
      ABET(b): An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution
      ABET(c): An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs
7. Brief list of topics to include but not limited to:
   * Internal organization of digital computers;
   * Assembly language programming;
   * Digital logic design;
   * Combinational circuits.
   * Overview and history of computer architecture
   * Combinational vs. sequential logic/Field programmable gate arrays as a fundamental combinational + sequential logic building block
   * Computer-aided design tools that process hardware and architectural representations
   * Physical constraints (gate delays, fan-in, fan-out, energy/power)
   * Bits, bytes, and words
   * Numeric data representation and number bases
   * Fixed- and floating-point systems
   * Signed and twos-complement representations
   * Representation of non-numeric data (character codes, graphical data)
   * Representation of records and arrays
   * Basic organization of the von Neumann machine
   * Assembly/machine language programming
   * Instruction formats
   * Addressing modes
   * Subroutine call and return mechanisms (cross-reference PL/Language Translation and Execution)
   * I/O and interrupts
   * Heap vs. Static vs. Stack vs. Code segments

1. Course: CSCI 3212 Computer Organization & Architecture
2. Credits: 3 Credits Contact Hours: 150 minutes per week
3. Course Coordinator: Dr. Gongbing Hong
4. Text: Computer Design and Architecture. 5th Ed. by Shiva
5. Specific course information
   a. Study of computer architecture. Topics include digital logic design, sequential and combinational circuits, micro-programming and CPU organization, memory addressing, internal bus organization, and control. Laboratory projects will include the physical electronic implementation of digital logic designs and the complete construction of an operational digital computer system.
   b. Prerequisites: C or better in CSCI 3211
   c. Required
6. Specific goals for the course
   a) Outcomes of instruction:
   1. Use CAD tools for capture, synthesis, and simulation to evaluate simple building blocks (e.g., arithmetic logic unit, registers, movement between registers) of a simple computer design.
   2. Explain the organization of the classical von Neumann machine and its major functional units.
   3. Describe the progression of computer technology components from vacuum tubes to VLSI, from mainframe computer architectures to the organization of warehouse-scale computers.
   4. Design the basic building blocks of a computer: arithmetic-logic unit (gate-level), registers (gate-level), central processing unit (register transfer-level), memory (register transfer-level).
   5. Evaluate the functional and timing diagram behavior of a simple processor implemented at the logic circuit level.
   6. Describe how an instruction is executed in a classical von Neumann machine, with extensions for threads, multiprocessor synchronization, and SIMD execution.

   b) Student Outcome

   SO 1: An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems.
   SO 2: An ability to apply design and development principles in the construction of software systems.
   SO 3: An ability to function effectively on teams to accomplish a common goal

   c) ABET Enabled Outcomes

   ABET (b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution
   ABET (c) An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs
   ABET (d) An ability to function effectively on teams to accomplish a common goal
7. Brief list of topics to include but not limited to:
   * Memory and Storage, Computer Org. and Programming;
   * Designing circuits with sequential logic;
   * Design of a complete computer architecture;
   * Laboratory implementation of a design of a complete computer architecture;
   * Input and output systems;
   * Advanced architectures.

1. COURSE: CSCI 3341, Operating Systems

2. Credit Hours: Three Contact Hours per week, 16 weeks per semester

3. Course Coordinator: JF Yao

4. Text: Operating Systems Internals and Design Principles- 7th edition, by William Stallings (ISBN-13: 978-0132309981)

SUPPLEMENT MATERIAL:

* Linux for Programmers and Users, by Graham Glass and King Ables (ISBN-13: 978-0131857483)
* Other online resources about Linux

5. Specific course information

a) A part of the thorough grounding in key principles and practices of computing is an in depth experience with the principles and designs of modern operating systems, from the operating system designer perspective. This course provides an introduction and foundation for achieving this objective

b) Prerequisites: C or higher in CSCI 3410 – Introduction to Data Structures

c) Required

6) Specific goals for the course

a) Outcomes of instruction

1. Students will demonstrate proficiency and mastery in basic principles, structure, and functions of operating systems and designs of modern operating systems
2. Students will be able to develop basic operating system components, such as process synchronization, processor scheduling, memory management, deadlocks handling, storage management
3. Student will understand the security issues in operating systems.

b) Student outcomes

SO 2: An ability to apply design and development principles in the construction of software systems

c) ABET Enabled Outcomes

ABET (b): An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution

ABET (c): An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs

ABET (e): An understanding of …, security… issues …

ABET (h): Recognition of the need for and an ability to engage in continuing professional development

ABET (i): An ability to use current techniques, skills, and tools necessary for computing practice.

ABET (k): An ability to apply design and development principles in the construction of software systems of varying complexity.

7. Course topics included but not limited to:

* Computer System Overview
* Processes
* Threads
* Concurrency: Mutual Exclusion and Synchronization
* Concurrency: Deadlock and Starvation
* Memory Management
* Virtual Memory
* Uniprocessor Scheduling
* Multiprocessor and real-time Scheduling
* I/O Management and Disk Scheduling
* File Systems
* Computer Security Threats
* Computer Security Techniques

1. CSCI3342 Networks and Network Programming

2. Credits: 3 credits Contact Hours: 150 minutes (75/75) 16 weeks per semester.
3. Course Coordinator: Gongbing Hong
4. Text: Computer Networks and Internets, 5th edition by Douglas E. Comer
5. Specific course information
    a. This course provides basic concept of the design and development of multitasking systems, client-server organizations, and distributed applications. Computer network design, implementations and protocols are included.
    b. Prerequisites: C or better in CSCI 3341
    c. Required

6. Specific goals for the course:

a) Course Learning Outcomes:

* Be able to explain how the internet works
* Understand the state-of-the-art in network protocols
* Understand network architectures and applications
* Be able to explain the principles of how to design network applications
* Be able to program network applications

b) Student Outcomes

SO1: An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems.
SO2: An ability to apply design and development principles in the construction of software systems.
SO3: An ability to function effectively on teams to accomplish a common goal.

c) ABET Enabled Outcomes

ABET (b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution
ABET (c) An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs
ABET (d) An ability to function effectively on teams to accomplish a common goal
ABET (h) Recognition of the need for and an ability to engage in continuing professional development
ABET (i) An ability to use current techniques, skills, and tools necessary for computing practice.
ABET (k) An ability to apply design and development principles in the construction of software systems of varying complexity.

7. Brief list of topics to include but not limited to:

* Inter-process Communication;
* Sockets;
* Client/server systems;
* TCP/IP Protocols and networks
* Computer networks, LANs, wans, protocols, models

1. COURSE: CSCI13410, Introduction to Data Structures

2. Credit Hours: 3 Credit Hours (75/75)

3. Course Coordinator: Dr. Gita Phelps

4. Text: Starting Out with Java – from Control Structures through Data Structures, by Tony Gaddis and Godfrey Muganda, (Third Edition) 2016

5. Specific course information:

a. This course serves the objective of providing a thorough grounding in the basic mathematical and scientific principles of algorithms and advanced software design techniques. This advanced treatment of programming principles, practices and experiences serves the objectives of preparation for more advance studies in Computer Science and preparation for professional programming employment.

b. Prerequisites: C or higher in CSCI 1302 (Computer Science II)

c. Required

6. Specific course information

a) Outcomes of instruction

1. Students will demonstrate proficiency and mastery in the following basic data structures and algorithms: Sorting and Searching, Lists, Stacks, Queues, Trees, Heaps and Priority Queues, Hashing

2. Students will be able to develop these data structures in Java, C++, or other programming languages from scratch.

3. Students will be able to analyze the complexity of a program using mathematical functions such as Big-O.

b) Student Outcome

SO 2: An ability to apply design and development principles in the construction of software systems.

c) AET Enabled Outcomes

ABET (b): An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution

ABET (c): An ability to design, implement, and evaluate a computer- based system, process, component, or program to meet desired needs

ABET (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 comprehension of the tradeoffs involved in design choices.

7. Course Topics Included But Not Limited To:

* Recursion	* Searching	* Queues
* Generic Types	* Algorithm Analysis (Big O)	* Trees
* Library Collections (including Sets, Maps, etc.)	* Array-Based Lists	* Heap
	* Linked Lists	* Priority Queues
* Sorting	* Stacks	* Hashing

COURSE: CSCI3680, Discrete Structures

2. Credit Hours: 4 FOUR Contact hours per week, 16 weeks

3. Course Coordinator: JF Yao

4. Text: "Mathematical Structures for Computer Science –Discrete Mathematics and Its Applications", seventh Edition, 2014, by Judith Gersting.

5. Specific course information
   a. This course serves the objective of learning an intensive introduction to discrete mathematics as it is used in computer science.
   b. Prerequisite: C or better in CSCI 1302 (Computer Science II) and in Math 1113 (Pre-Calculus).
   c. Required

6. Specific goals for the course

a) Outcomes of instruction:

Students will be able to:

1. explain with examples and perform the operations associated with sets, functions, and relations.
2. demonstrate basic counting principles
3. apply formal methods of symbolic, propositional, and predicate logic.
4. use formal logic proofs and logical reasoning to solve problems.
5. relate the ideas of mathematical induction to recursion and recursively defined structures.
6. apply the tools of probability to solve problems such as the average case analysis of algorithms, and hashing.
7. understand cryptography
8. understand basic graph and Trees
9. understand basic Finite-State Machines, Turing Machines, and Formal Languages

b) Student Outcomes

SO 1: An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems.

c) ABET Enabled Outcome

ABET (a) An ability to apply knowledge of computing and mathematics appropriate to the discipline

ABET (b): An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution

ABET (e): An understanding of security issues

ABET (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 comprehension of the tradeoffs involved in design choices.

1. Course CSCI 4320 Software Engineering

2. Credits: 3 hrs Contact Hours: 150 each week (50/50/50)

3. Course Coordinator: Dr. Gita Phelps

4. Text: Object-Oriented and Classical Software Engineering by Stephen R. Schach (2010, Hardcover)
ISBN-10: 0073376183 | ISBN-13: 9780073376189

5. Specific course information

a. An introduction to software engineering methodologies addressing each phase in the life cycle of software. Programming assignments use modern software engineering tools and languages

b. Prerequisite: C or better in CSCI 4710 and Senior status.

c. Required

6. Specific goals for the course

a. Course learning outcomes

1. Students will be able to apply modern software engineering tools and languages;
2. Students will be able to develop a complete software package while addressing phases in the software development life cycle;
3. Students will be able to work in teams in both classroom and laboratory environments;
4. Students will demonstrate an ability to communicate effectively.
5. Students will be able to identify different design patterns commonly occurring in software engineering.

b) Student Outcomes

SO 1: An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems.
SO 2: An ability to apply design and development principles in the construction of software systems.
SO 3: An ability to function effectively on teams to accomplish a common goal.
SO 5: An ability to communicate effectively with a range of audiences

c) ABET Enabled Outcomes

ABET (b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution
ABET (c) An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs
ABET (d) An ability to function effectively on teams to accomplish a common goal
ABET (f) An ability to communicate effectively with a range of audiences
ABET (h) Recognition of the need for and an ability to engage in continuing professional development
ABET (i) An ability to use current techniques, skills, and tools necessary for computing practice.
ABET (k) An ability to apply design and development principles in the construction of software systems of varying complexity.

7. Brief list of topics to include but not limited to:

* Software engineering methodologies addressing each phase in the life cycle of software development.
* UML Documents for Object Oriented Analysis
* Reusability & Portability
* Version Control Issues
* Good Programming Practices
* Unit Testing
* Implementation & Integration Issues
* Quality Assurance Issues
* Post Delivery Maintenance

1. CSCI4330 Programming Language Design and Survey

2. Credits: 3 credits Contact Hours: 3 credit hours per week, 16 weeks per semester.

3. Course Coordinator: Dr. Yi (Cathy) Liu

4. Text: Programming Language Progmatics by Michael L. Scott Lisp Textbook: http://www.htdp.org/2003-09-26/Book/curriculum.html Prolog Textbook: Adventure in Prolog Python Textbook: Bite of Python

5. Specific course information

This course provides an introduction to the foundations of programming languages, emphasizes language design concepts, allows a thorough understanding of programming language semantics and critical implementation characteristics. This provides the basis for the best choice and use of existing languages, as well as language design.

a. Prerequisites: C or better in CSCI 3410

b. Elective

6. Specific goals for this course:

a) Course Learning Outcomes:

1. Understand various advanced programming constructs and idioms.
2. Discuss how various advanced programming constructs interact with the definition and implementation of other language features.
3. Understand static type and dynamic type systems
4. Compare and contrast (1) the procedural/functional approach (defining a function for each operation with the function body providing a case for each data variant) (2) the object-oriented approach (3) logic programming
5. Be able to write functional programs
6. Be able to write logic programs
7. Be able to write scripting programs

b) **Student Outcomes**

SO 2: An ability to apply design and development principles in the construction of software systems.

c) ABET Enabled Outcomes

ABET (b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution

ABET (c) An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs

ABET (i) An ability to use current techniques, skills, and tools necessary for computing practice.

7. Brief list of topics to include but not limited to:

* Various advanced programming constructs and idioms.
* Discuss how various advanced programming constructs interact with the definition and implementation of other language features.
* Static type and dynamic type systems
* Identify program properties checked statically and program properties checked dynamically
* Compare and contrast (1) the procedural/functional approach (defining a function for each operation with the function body providing a case for each data variant) and (2) the object-oriented approach (3) logic approach
* Lisp
* Prolog
* Python

1. CSCI4520 Design & Analysis of Algorithms

2. Credits: 3 credits Contact Hours: 3 credit hours per week, 16 weeks per semester.

3. Course Coordinator: Dr. Yi (Cathy) Liu

4. Text: Introduction to the Design & Analysis of Algorithms by Anany Levitin

5. Specific course information

a. This course provides basic concept of Algorithms Analysis; Fundamental computing algorithms, algorithmic strategies (brute-force algorithms, divide-and-conquer, decrease-and-conquer, Transform-and-conquer, Greedy algorithm, Dynamic Programming), and basic computability.

b. Prerequisites: C or better in CSCI 3410 and CSCI 3680

c. Required

6. Specific goals for the course

a) Course Learning Outcomes:

1. Understand each of the strategies (brute-force, greedy, divide-and-conquer, and dynamic programming)
2. Be able to Determine an appropriate algorithmic approach to a problem
3. List and contrast standard complexity classes.
4. Perform empirical studies to validate hypotheses about runtime stemming from mathematical analysis. Run algorithms on input of various sizes and compare performance.
5. Give examples that illustrate time-space trade-offs of algorithms.
6. Explain the use of big omega, big theta, and little o notation to describe the amount of work done by an algorithm.
7. Use recurrence relations to determine the time complexity of recursively defined algorithms.
8. Solve problems using graph algorithms
9. Provide examples of classic NP-complete problems

b) Student Outcomes

SO1: An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems.

SO2: An ability to apply design and development principles in the construction of software systems.

SO3: An ability to function effectively on teams to accomplish a common goal.

c) ABET Enabled Outcomes

ABET (a) An ability to apply knowledge of computing and mathematics appropriate to the discipline

ABET (b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution

ABET (d) An ability to function effectively on teams to accomplish a common goal

ABET (f) An ability to communicate effectively with a range of audiences

ABET (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 comprehension of the tradeoffs involved in design choices.

7. Brief list of topics to include but not limited to:

* Brute-force programming
* Greedy programming
* Space-time trade off
* Divide-and-conquer
* Dynamic programming
* Big omega, big theta, and big o notation
* Recurrence relations for algorithms.
* Tree, graph algorithms
* Classic NP-complete problems

1. COURSE: CSCI4710, Databases

2. Credit Hours: Three contact hours per week, 16 weeks per semester

3. Course Coordinator: JF Yao

4. Text : Database Systems, by Thomas Connolly and Carolyn Begg
Hands-on Oracle Database 10g Exp Ed for Windows, by Bobrowski
SUPPLEMENT MATERIAL:* Online resources about Databases

5. Specific course information

a. An introduction to basic concepts, design techniques, and theories in database management systems. The primary emphasis is on the relational database management systems. The goal is for students to be able to design and maintain a relational database management system as a DBA after taking this course.

b. Prerequisite: C or above in CSCI 3410 - Data Structures

c. Required

6. Specific goals for the course

a) Outcomes of instruction

1. Students will become acquainted with basic principles, structure, and functions of database management systems.
2. Students will be able to use SQL comfortably.
3. Students will be able to develop a relational database on a real world DBMS.
4. Students will be able to refine schema and to tune a DBMS
5. Students will be able to create and modify storage structures and access methods

b) Student Outcome

SO 2: An ability to apply design and development principles in the construction of software systems.
SO 3: An ability to function effectively on teams to accomplish a common goal.

c) Enabled Abet Student Outcomes:

ABET (b): An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution
ABET (c): An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs
ABET (d): An ability to function effectively on teams to accomplish a common goal
ABET (h): Recognition of the need for and an ability to engage in continuing professional development
ABET (i): An ability to use current techniques, skills, and tools necessary for computing practice.
ABET (k): An ability to apply design and development principles in the construction of software systems of varying complexity.

7. COURSE TOPICS INCLUDED BUT NOT LIMITED TO:

* Introduction to Databases
* The Relational Model
* Relational Algebra
* Relational Calculus
* SQL: Data Manipulation & Data Definition
* Advanced SQL
* Query-By-Example (QBE)
* Schema Refinement and Normal Forms
* Advanced Normalization
* Conceptual Database Design
* Logical Database Design for the Relational Model
* A Real World DBMS: Oracle
* Oracle SQL
* PL/SQL
* Web Programming with PL/SQL