Computer Science Department
cs.salemstate.edu

ITE 105 Problem Solving with Algorithms **3 cr.**

Catalog description:

This course serves as an introduction to programming. Using flow charts, pseudo-languages, and software development strategies, students will learn techniques for identifying and selecting solutions to problems by designing algorithms, using stepwise refinement and structured programming techniques. Students will design algorithms using pseudo-code, implement algorithms using a simplified programming environment, and participate in hands-on debugging, testing, and documenting activities. Topics include principles of programming, the logic of constructing a computer program, integrating modules into a cohesive application, and fundamentals of programming languages. In-class exercises allow students to practice these techniques while solving assigned problems. Three lecture hours per week.

Prerequisites: High school algebra I & II plus experience with a window-based operating system and the use of email and a word processor. Recommended for students with no prior programming experience.

Course Narrative:

Computing disciplines learn by experience. Problem solving is an important aspect of human learning. Being exposed to different problem-solving techniques, this problem solving ability can then be taken to the next level, which is actually learning a language. Learning the techniques of writing algorithms and flowcharts will serve as an initial step in the process of learning how to program.

The emphasis of this class is on modern concepts and programming conventions in order to build a solid foundation in programming logic. This course presents the fundamental techniques of designing and writing algorithms, flowcharts, and pseudo-codes. The concepts of algorithms, flowcharts, and pseudo-codes will first be dealt in order to solve real life problems but without needing to learn the syntax of any specific programming language. Testing the algorithms for verification and correctness is another important aspect students need to learn as the course proceeds. Major topics include principles of programming, the logic of constructing a computer program, integrating modules into a cohesive application, testing for correctness, and fundamentals of programming languages.

A major course objective is for students to be able to demonstrate the ability to break a large task into modules and be able to implement designed algorithms in an executable format. This objective is accomplished by taking examples from multiple programming languages to make students realize how universal problem solving can be implemented using the syntax of other languages. Thus, the objective of problem solving is accomplished along with testing abilities of how to argue the correctness and completeness of algorithms.

Goals:

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

G1: use abstract thinking and formal logic in implementing algorithms for solving a wide variety of problems;

G2: use standard methods of problem analysis to design problem solutions and appropriate

techniques (flow charts, pseudo-languages) to represent solutions;

G3: use a simplified subset of a programming language to solve several classic problems;

G4: describe formal methods of problem solving, characteristics of different classes of algorithms, and fundamental properties of programming languages;

Course Objectives:

Upon successful completion of the course, a student will have demonstrated the ability to:

O1: apply correct terminology when analyzing problems and designing solutions for different classes of problems;

O2: solve problems by applying problem solving techniques and presenting solutions as algorithms using multiple algorithm description methods;

O3: argue correctness, effectiveness, and completeness of developed algorithms;

O4: implement designed algorithms in an executable format and use verification techniques to argue program correctness;

O5: demonstrate ability to break a large task into modules, develop and use APIs, and work in groups to develop a larger task using modular technique;

Program Objective / Course Objective matrix (For ABET Accreditation Purposes)

(The following Matrix maps the Program Objectives for Information Technology Program outlined by Accreditation Board of Engineering Technology (ABET) with the Course Objectives. The check marks below the course objectives represent that those course objectives accomplish specific program objectives set forth by ABET. The program objectives that have a * in front of them means that that course does not address those program objectives.)

Program Objective	O1	O2	O3	O4	O5
PO-A: An ability to apply knowledge of computing and mathematics appropriate to the program's student outcomes and to the discipline.	✓		✓	✓	
PO-B: An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.	✓	✓	✓		
***PO-C:** An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.					
PO-D: An ability to function effectively on teams to accomplish a common goal.					✓
***PO-E:** An understanding of professional, ethical, legal, security and social issues and responsibilities.					
PO-F: An ability to communicate effectively with a range of audiences.					✓
***PO-G:** An ability to analyze the local and global impact of computing on individuals, organizations, and society.					
***PO-H:** Recognition of the need for and an ability to engage in continuing professional development.					

Program Objective	O1	O2	O3	O4	O5
PO-I: An ability to use current techniques, skills, and tools necessary for computing practice.	✓	✓			
PO-J: An ability to use and apply current technical concepts and practices in the core information technologies.	✓	✓			
***PO-K:** An ability to identify and analyze user needs and take them into account in the selection, creation, evaluation and administration of computer-based systems.					
***PO-L:** An ability to effectively integrate IT-based solutions into the user environment.					
PO-M: An understanding of best practices and standards and their application.	✓	✓		✓	✓
***PO-N:** An ability to assist in the creation of an effective project plan.					

Topics:

The column on the right hand side represents the Body of Knowledge and number of hours (in parenthesis) set forth by ABET accreditation board for accomplishing minimum required hours assigned for different categories. More information on this body of knowledge can be found in Appendix A "The IT Body of Knowledge" on Page 68 of the following document.
http://www.acm.org//education/curricula/IT2008%20Curriculum.pdf

- Introduction to Computers and Programming **ITF(1), ITF2(1), SP1(1)**

- Input, Processing, and Output **PF1(2), PF2(1), PF4(2.5)**
  - Algorithm, Definition of Algorithm
  - Basic control flow and data types
  - Problem-solving strategies for algorithm development
  - Types of algorithms and analysis of different algorithms
  - Different ways of representing an algorithm, Flowcharts, Pseudo-code

- Modules, Decision Structures, and Boolean **PF1(1), PF2(2), PF4(2)**
  - Introduction to Modules
  - Defining and Calling a Module
  - Local Variables and Passing Arguments
  - Introduction to Decision Structures
  - Boolean algebra
  - Basic operations: arithmetic, logical, and control operations
  - Example of using algorithms for problem solutions

- Repetition Structures and Functions **PF1(1), PF2(2)**

  - Introduction to Repetition Structures

- Condition controlled and Count Controlled Loops
- Functions

- Testing and Input Validation **SIA5(2)**
- Arrays, Sorting and Searching Arrays **PF1(1), PF2(2)**

- Inventing a programming language **PT1(0.5), PT2(0.5), PT3(2), PF1(1), PF3(4.5), PF4(2), IPT7(2)**

  - Formalize PSL
  - Algorithm design vs. programming
  - Path to a programming language
  - Recording algorithms using programming language
  - System Services vs. programming language constructs

- Programming languages as a part of Computing Environment **IPT2(2), IPT4(2), IPT5(1), IPT7(1)**
  - Compilers
  - Interpreters
  - Scripting (types of languages and examples)
  - Software Security Practices
  - Programming environments:
    - System programming
    - Application programming
    - Web programming

The emphasis of this course is on understanding and exercising problem solving using algorithms, algorithm development, and algorithm representation techniques. Properties and characteristics of classes of programming languages are analyzed, followed by a discussion of how programming languages are used to encode algorithms and on how an algorithm, recorded using a programming language, becomes an executable program. Overall software development cycle is discussed step-by-step – from a design stage to verification and testing of the executable program.

Student Experiences:

Organization of the course

The course consists of lectures, in-class exercises, homework assignments, quizzes, and two exams – a midterm and a final. In addition to the textbook, Internet resources are widely used during the course by the instructor as well as students.

In-class exercises:

In-class exercises may consist of:

- Discussions of the material presented during the lecture
- Exercises in problem-solving with algorithms using techniques discussed during lectures
- Exercises in using pseudo-code language and simplified elements of programming languages for encoding algorithms
- Exercises designed to better understand different stages of a software development process

Group discussion time and presentations will be scheduled as a part of lectures to ensure student's involvement and active participation in the process.

Assignments

Homework assignments, given weekly, will exercise theoretical principles discussed during the lectures through practical exercises. Assignments require students to use information given during the lectures and textbooks, and perform Internet research for necessary materials. Regular writing assignments include but are not limited to:

- exercising material presented during lectures (including more in-depth work on examples used during lectures);
- review of textbooks and technical articles as a part of research assignments;
- written presentations of research findings;
- proposals to solve problems formulated by the instructor using algorithms;

Specific requirements for each assignment will be stated when the assignment is distributed; all written submissions will be graded against the Writing rubric. Presentations will be assessed based on the Presentation rubric.

Quizzes, Tests, and Examinations

There will be quizzes, homework assignments, in class exercises, a midterm, and a cumulative final. Quizzes and exams will include multiple choice and problem solving tasks.

Grading

Final grades will be structured on the basis of the following approximate weights:

- In-class exercises 10%
- Homework assignments 25%
- Quizzes 25%
- Midterm exam 20%
- Final exam 20%

Student Experiences by Course Outcome (Objective) matrix:

	O1	O2	O3	O4	O5
In-class exercises	✓	✓	✓	✓	✓
Homework assignment	✓	✓	✓	✓	
Quizzes	✓		✓		
Midterm exam	✓	✓	✓		
Final Exam	✓		✓	✓	✓

Tools and Web resources:

- Problem Solving with Algorithms http://faculty.kfupm.edu.sa/ics/alfy/files/teaching/ics201/11.ppt
- Online Boolean algebra calculator http://joshtam.net/world/bee-calc.html
- Von Newmann computer model http://www.c-jump.com/CIS77/CPU/VonNeumann/lecture.html
- Discussion using images of Von Newmann model http://www.google.com/search?q=Von+Neumann+computer+model&hl=en&complete=0&site=webhp&prmd=imvns&tbm=isch&tbo=u&source=univ&sa=X&ei=3mPKT6npM8TX0QHP6rCaAQ&ved=0CGAQsAQ&biw=1207&bih=576
- Discussion using images of “Software development cycle”

http://www.google.com/search?q=software+development+cycle&hl=en&complete=0&site=webhp&prmd=imvns&tbm=isch&tbo=u&source=univ&sa=X&ei=yXDKT8avFM3l6gG7vugz&ved=0CHgQsAQ&biw=1207&bih=576

Bibliography:

- Gaddis, Tony. **Starting Out with Programming Logic and Design**. Fourth Edition. Pearson, 2015.
- Gaddis, Tony. **Starting Out with Programming Logic and Design**. Third Edition. Pearson, 2012.
- Backhouse, R. **Algorithmic Problem Solving**. First Edition. Wiley, 2011.
- Farrell, Joyce. **Programming Logic and Design: Introductory**. Seventh Edition, Joyce Farrell, 2013.
- Dale, Nell; Lewis, John. **Computer Science Illuminated**. Fifth Edition. Jones and Bartlett Publishers, Inc, 2012.
- Miller, B.N. **Problem Solving with Algorithms and Data Structures Using Python**. Second Edition, Franklin Beedle & Associates, 2011.