Catalog Description
Programming emphasizing recursion, data structures, and data abstraction. Elementary analysis of and reasoning about programs. Public policy issues. Extensive programming. Three hours of lectures and one hour of computer laboratory.

Course Objectives
This course introduces many of the core concepts of computer science, as well as touching on a few pieces of its history. An important goal of this course is to expand students’ view of the computing landscape by helping them explore and experiment with options beyond what they’ve already seen. The course uses the programming language Haskell to emphasize two main themes: patterns and problem solving. The goal is to help students think about programming in a more systematic and logical way and to recognize core concepts that they take with them to other courses and into their professional lives.

Prerequisites
MAT 295 is a co-requisite, as an indication of mathematical maturity.

Course Outcomes
After completion of the course, students should be able to:

- Use the Unix command-line interpreter to perform basic tasks, including: managing files and directories, changing file permissions, checking a print queue, printing files (both postscript and non-postscript), checking available options for a Unix program, and using basic pipes and filters. [ABET (a), (i)]
- Recall the 4 steps of Polya’s problem-solving method. [ABET (b)]
- When given an informal English specification of a problem, extract a relevant collection of abstract types from that specification (i.e., the contract). [ABET (b), (c), (j)]
- When given a formal specification, construct concrete examples that satisfy that specification.
- When given a problem specification, recognize, comprehend, and apply the concepts of pattern matching and higher-order functions to design programs. [ABET (b), (c), (j)]
- When given an informal but fairly precise English description, write a Haskell program that accurately captures the desired behavior. [ABET (a), (b), (c), (i), (j), (k)]
- When given a Haskell expression, determine and explain whether or not it is well-typed. [ABET (b)]
- Read, write, and understand programs written in the functional paradigm (particularly, in the language Haskell) [ABET (a), (c), (i), (j), (k)]
- When given a moderate-sized Haskell program and relevant input, calculate the result of that program. [ABET (a)]

Outcome Measurement
Outcomes are assessed through a combination of weekly homework assignments, weekly labs, and in-class exams or quizzes.
Course Topics

CAC Category Content
1 Data Structures
1 Algorithms
1 Software Design
0 Computer Organization & Architecture
1 Programming Languages