**CSE 6010 / CX 4010 Computational Problem Solving for Scientists and Engineers**

2-3-3 (2 hours of lecture per week, 3 hours of unscheduled lab work (programming), 3 hours credit) Credit not given for Computer Science or Computational Media students.

Prereq: CS 1371 and senior or graduate standing in a mathematics, science, or engineering program (juniors by permission of instructor)

**Overview**

This course is designed for advanced undergraduate and graduate students with limited computing background to prepare them for upper division and graduate coursework in computational science and engineering. Development of computer software to solve problems in science and engineering. Computing principles, computer architecture, algorithms and data structures; software development methodologies and tools; introduction to advanced topics such as parallelism and object-oriented design; Students will be expected to devote a significant amount of time to practice and develop computer programming skills.

Starting from real-world problems that commonly arise in science and engineering, students will develop the knowledge and skills needed to develop computational solutions to attack these problems.

Students shall demonstrate proficiency is solving typical computational problems that arise in science and engineering through the completion of a series of assignments requiring the development of new software. Examinations for the course emphasize testing the proficiency of students to developing working software.

Topics include:

* Computational problems in science and engineering
* Computer architecture necessary to understanding program behavior and performance
* Elementary data structures, including arrays, lists, trees, graphs, hash tables, heaps
* Algorithms and their analysis, including dynamic programming and concepts of  complexity
* Computer programming
* Software engineering methodologies (best practices for code design, debugging, testing,  documentation) and software development tools such as integrated development environments (IDEs), revision control, project/bug tracking (e.g., bugzilla), scripting and related tools such as Matlab and python
* Advanced programming concepts such as data abstraction and objects, object-oriented design, parallel computing  **Additional assignment and topics for graduate students:** 
  + There will be additional assignments, such as research paper readings for topics covered in the class, and/or additional readings for each topic.
  + Programming assignments will also include additional questions that are more open-ended (less structured).