CSCI 1913 - Introduction to Algorithms, Data Structures, and Program Development
(4.0 cr; [=1902]; Prereq-1103 or 1113 or #; fall, spring, summer, every year)
Builds on prerequisite courses, presenting advanced object-oriented
programming to implement abstract data types (stacks, queues, linked lists,
hash tables, binary trees) using the Java language.
Searching and sorting algorithms. Basic algorithmic analysis.
Introduction to scripting languages using the Python language.
Substantial programming projects, integral weekly lab.

SLO
Student in the course:

- Can identify, define, and solve problems

Please explain briefly how this outcome will be addressed in the course. Give brief examples of class work related to the outcome.

CSci 1913 is focused on the use of algorithms and data structures to solve problems. Specifically,the course labs, homework assignments, and exams all ask the students to solve various problems using appropriate software design methods and software tools. For example, students need to decide which problem solving strategies (such as divide and conquer) might be useful for a specific problem, construct a solution, design appropriate data types and algorithms, and verify the correctness of the solution.

How will you assess the students' learning related to this outcome? Give brief examples of how class work related to the outcome will be evaluated.

This SLO will be assessed through labs, homeworks, exams, and in-class exercises. Each of these types of students' work is problem-based; most involve open-ended problems where students need first to identify, define, and/or clarify what the problem is before solving it, and explain the solution they propose.

Sample syllabus

Class Format

4 credits, 3 large class + 2 lab hours per week. The lab is used to design, write, and test programs on various subjects.

CSci 1913 is the second Computer Science class in the major. It is for students who have taken a college level course in C/C++ (CSci 1113) or in Java (Csci 1103), or have AP credits. The course focuses on learning object-oriented concepts (class, object, method, inheritance) and abstract data types (stack, queue, binary tree, etc) using the Java language. In addition the course introduces students to the use of modern debugging and software development tools, and to scripting languages using the Python language.

1913 is a required undergraduate course which students should take in their freshman or sophomore year. 1913 is one of the courses required for admission to the CSci major, and is a prerequisite for many of the higher level CSci classes.

Classes having 1913 as a Prerequisite include CSci 2021 (Computer Organization), 3081 (Program Design and Development), and 4041 (Algorithms and Data Structures), which in turn are prerequisites for many other upper division classes. 1913 teaches fundamental abstract data types (stacks, queues, and trees), which are used extensively in the rest of Computer Science. To give a few quick examples, (i) the queue abstract data type is used for CPU and disk scheduling; (ii) stacks are used extensively in the compilers, architecture and programming languages courses to describe parsing algorithms for arithmetic expression, computer programs etc.; (iii) abstract data types and object oriented programming are used as basic tools for software development.

Specifically the class covers:
* Object-oriented design
  - Sorting and searching algorithms
  - Basic complexity analysis
* Definition of classes: fields, methods, and constructors
* Subclasses, inheritance, and method overriding
* Abstract data types and their implementation (stacks, queues, priority queues, linked lists, binary trees)
* Strategies for choosing the appropriate data structure
* Modern programming environments
  Programming using library components and their APIs
* Debugging strategies
* Documentation and program style

Upon successfully completing this class, students should be able to use abstract data types in solving a variety of problems, and be able to implement their solutions in an object oriented programming language. Specifically, students should be able to:
1. identify which abstract data type and/or algorithm could be useful in representing or solving a given problem, and why;
2. develop a functioning object oriented program in Java, and document its functionality in terms of the given problem.
3. construct, execute and debug programs using a modern IDE and associated tools, such as unit testing tools and visual debuggers.
4. compare implmentations of a variety of searching and sorting algorithms with respect to complexity.
5. apply a variety of strategies to the testing and debugging of programs.
6. apply consistent documentation and program style standards that contribute to the readability and maintainability of software.
7. modify or specialize abstract data types to adapt them to problems that are not amenable to straightforward use of the abstract data types;
8. compare alternative implementations of data structures with respect to performance;
9. develop applications requiring file I/O and use of libraries using a scripting language.

Weekly outline of material covered

Week 1: Overview. Introduction to Java and to IDE.
Week 2: Object oriented design.
Week 3: Inheritance.
Week 4: Linked lists. Stacks,
Week 5: Queues. Priority queues.
Week 6: Binary search trees. Heaps.
Week 7: Hash tables.
Week 8-9: Searching and sorting algorithms.
Week 10-11: Introduction to Python: syntax and data types
Week 12-13: Object-oriented programming in Python
Week 14-15: Abstract data types in Python.

Grading:

15% In-class Exercises
6% Quizzes
12% Midterm Exam 1
12% Midterm Exam 2
20% Final
25% Labs
10% Homeworks

Using the above weights, a total of 90% and up will earn you some level of A, 80% and up at least some level of B, 70% and up at least some level of C, 60% and up at least a D.

Class Webpage: All future handouts, assignments, announcements, and any additional material will be available through the 1913 class web page in the directory http://www.cselabs.umn.edu/classes/

Labs: Weekly labs include a variety of problems designed to help you acquire programming adcanced programming skills in the Java and Python languages and to develop good software design practices.

Scholastic Conduct: Although you are free to discuss assignments with others, the work you submit for grading must represent your own efforts only. Exams are closed book and note and are to be completed using only your own knowledge of the course material. The experience gained from the labs will be very helpful for the exams and future labs may build on previous ones, so put in the effort needed to fully understand the solutions. Cheating on quizzes or exams is a serious offense, and will be dealt with as such. Additionally, labs are done in groups of two, but collaboration with others outside your group of two is prohibited. Homework assignments are individual efforts. You may discuss (in a general way) labs and homework problems with others, but you may not collaborate by writing code or specific solutions with others (with the exception of your lab partner in the case of labs). Copying other's answers, or letting another person copy your answers (either intentionally or as a result of negligence) is a serious situation and can result in failing the course. For further information on academic misconduct please see http://www-users.cs.umn.edu/ barry/intro/acad-conduct.html. If you have any questions about what is and is not allowable in this class, please ask the course instructor.

Incompletes: Incompletes will be given only in very rare instances when an unforeseeable event causes a student who has completed all the coursework to date to be unable to
complete a small portion of the work (typically the final assignment or exam). Incompletes will not be awarded for foreseeable events including a heavy course load or poorer-than-expected performance. Verifiable documentation must be provided for the incomplete to be granted, and arrangements for the incomplete should be made as soon as such an event is apparent.