Data Structures (CS 1520) Spring 2019

Time and Place: 8 - 9:15 AM Tuesday and Thursday in ITT 328 **and** either:

- Section 01: 8 – 9:50 AM Wednesday in **Wright 112**, or
- Section 02: 10 – 11::50 AM Wednesday in **Wright 112**.

Web-site: **http://www.cs.uni.edu/~fienup/cs1520s19/** (Lecture videos posted after class)

Class Email List: Send messages to Google group for the course at CS-1520-01-spring@uni.edu or CS-1520-02-spring@uni.edu

Instructor: Mark Fienup (fienup@cs.uni.edu)
Office: ITT 313
Phone: 319-273-5918 (Home 319-266-5379)
Office Hours: M: 8-11:45, 1:10-2, T: 9:30-11:45, 1:10-2, W: 1:10-2, Th: 9:30-11:45, 1:10-2, F: 8-10

Prerequisite: Intro. to Computing (CS 1510), and pre- or corequisite Discrete Structures (CS 1800)

Student Learning Outcomes: After this course you should be able to:
1. implement efficient data structures include: stacks, queues, lists, heaps, hash tables, trees, and graphs
2. design and implement "medium" sized programs using functional decomposition and be able to select appropriate data structures.

Text: *Problem Solving with Algorithms and Data Structures Using Python*, 2nd edition, by Bradley N. Miller and David L. Ranum. Franklin, Beedle & Associates. ISBN: 978-1-59028-257-1 (http://www.pythonworks.org/pythonds)
Free on-line version of the textbook: http://interactivepython.org/courselib/static/pythonds/index.html

Assignments: Assignments will consist of weekly laboratory exercises along with concurrent weekly or bi-weekly programming assignments.

Pedagogic Approach: In class, I'll tend to break up the lecture with active (and group) learning exercises to aid learning. While this is not formally graded, part (5%) of your grade will be based on your participation in (and attendence for) these in-class activities. Students benefit by (1) increased depth of understanding, (2) increased comfort and confidence, (3) increased motivation, and (4) being better prepared to work in groups on the job. This might sound great, but it will require you (and me) to work differently to prepare for class. Before the class, you must read the assigned reading, thought about what I asked you to think about, etc.; otherwise you won't be able to effectively participate during class.

Grading policy: There will be three tests (including the final). Tentative test dates and weighting of course components are:

Attendance and In-class Work:	5 %	
Labs:	15 %	
Programming Assignments:	20 %	
In-class Test 1:	20 %	(Thursday, February 21)
In-class Test 2:	20 %	(Thursday, April 4)
Final:	20 %	(Tuesday, May 7 from 8 - 9:50 AM in ITT 328)

Grades will be assigned based on straight percentages off the top student score. If the top student's score is 92%, then the grading scale will be, i.e., 100-82 A, 81.9-72 B, 71.9-62 C, 61.9-52 D, and below 52 F. Plus and minus grades will be assigned for students near cutoff points.

Scholastic Conduct: You are responsible for being familiar with the University' Academic Ethics Policies (http://www.uni.edu/pres/policies/301.shtml). Copying from other students is expressly forbidden. Doing so on exams or assignments will be penalized every time it is discovered. The penalty can vary from zero credit for the copied items (first offense) up to a failing grade for the course. If an assignment makes you realize you don't understand the material, ask questions designed to improve your understanding, *not* ones designed to discover how another student

solved the assignment. The solutions to assignments should be **individual, original** work unless otherwise specified. Remember: discussing assignments is good. Copying code or test-question answers is cheating.

Any substantive contribution to your assignment solution by another person or taken from a publication (**or the web**) should be properly acknowledged in writing. Failure to do so is plagiarism and will necessitate disciplinary action. In addition to the activities we can all agree are cheating (plagiarism, bringing notes to a closed book exam, texting during an exam, etc.), assisting or collaborating on cheating is cheating (e.g., supplying code for another student). Cheating can result in failing the course and/or more severe disciplinary actions.

Student Accessibility Services Statement:
The University of Northern Iowa (UNI) complies with the Americans with Disabilities Act Amendments Act of 2008 (ADAAA), Section 504 of the Rehabilitation Act of 1973, the Fair Housing Act, and other applicable federal and state laws and regulations that prohibit discrimination on the basis of disability. To request accommodations, it is the policy of the University for students with disabilities to register with Student Accessibility Services (SAS). UNI faculty are not obligated to provide accommodations for students with disabilities without proper notification from SAS and the student. Students may initiate the accommodation process at any time. However, accommodations are not retroactive, and the registration process takes time. Therefore, SAS staff always recommends that students initiate the process as soon as possible rather than wait for academic and social-emotional responsibilities to become overly stressful and/or overwhelming. Please contact SAS, located at ITTC 007, for more information either at (319) 273-2677 or accessibilityservices@uni.edu

Data Structures Schedule Spring 2019

Lect #	Tuesday		Thursday	
1	1/15	Ch. 1: Python Review; Classes	1/17	Ch. 2: Algorithm Analysis; Big-oh; timing
3	1/22	Ch. 2: Performance of Python Built-in data structures	1/24	Ch. 3: Linear data structures; stack implementations
5	1/29	Stack applications; Queue implementations	1/31	Queue applications; Deque
7	2/5	Ch. 6.6: Priority Queue: binary heap implementation	2/7	Unordered Lists implementations
9	2/12	Ordered List implementation Ch. 4: Recursion; Run-time stack	2/14	Recursion examples: Coin-change problem
11	2/19	Review for Test 1	2/21	**Test 1**
13	2/26	Backtracking v.s. dynamic programming coin-change problem	2/28	Ch. 5: Searching: linear and binary search Hashing: Open-address with rehashing
15	3/5	Chaining/closed-address hashing Dictionary implementations	3/7	Open-address implementation; Simple sorts: bubble, selecton, and insertion sorts
17	3/12	Advanced sorts: heap, merge and quick sorts	3/14	Ch. 6: tree terminology, binary tree implementation, parse tree application, and tree traversals
19	3/26	Binary Search Tree implementation	3/28	Binary Search Tree delete method
21	4/2	Review for Test 2	4/4	**Test 2**
23	4/9	AVL height-balanced trees	4/11	File structures vs. In-memory Data Structures B+ trees
25	4/16	Ch. 7: Graph terminology, traversals (BFS & DFS) Graph implementations	4/18	Select Graph algorithms: topological sort and Dijkstra's algorithm
27	4/23	Select Graph algorithms: Prim's algorithm; Modifications to BinHeap	4/25	Select Graph algorithms: TSP Backtracking and Best-first search Branch-and-bound
29	4/30	TSP: Approximation algorithm	5/2	Review for Final Exam
Final: **8:00 - 9:50 AM Tuesday, May 7 in ITT 328**				