Syllabus – CS 271 – Data Structures (4 credits) – Fall 2016

INSTRUCTOR: Tom Naps

OFFICE: Halsey 214, phone 424-1388

EMAIL: naps@uwosh.edu

MEETING TIME: MTWR 8:00 - 9:00

OFFICE HOURS: MW 9:30 - 11:30, Tuesday 11:30 - 1:00, Thursday 12:30 - 1:30

REFERENCES:

- Class lecture notes, labs, assignments available on D2L
- An online interactive textbook that is part of the OpenDSA project as hosted at Canvas. You have already received an email with instructions about how to enroll in the Canvas-hosted CS 271 course and access the book at Canvas. In case you've misplaced that email, you can use the following direct link to enroll in the course – `https://canvas.instructure.com/enroll/4RF8CW`

Content

- Algorithm Analysis for non-recursive algorithms
- Recursion, algorithm analysis for recursive algorithms, backtracking, algorithm design
- Building effective applications using linear data structures
  - Arrays/vectors
  - Linked lists
  - Stacks
  - Queues
- Higher-powered sorting algorithms and lower bounds on how good they can become
- Trees, binary search trees, heaps, other applications of trees
- Hash tables
- Graphs and their applications

Learning Outcomes

1. Given a non-recursive algorithm, you will be able to examine its loop structures and infer its asymptotic run-time using big-O notation.
2. Given a recursive algorithm, you will be able to examine its recursive structure, determine the corresponding recurrence relation (from a small collection of commonly occurring recurrence relations), and use the recurrence relation in determining the asymptotic run-time of the algorithm using big-O notation.
3. Given the description of a computational problem requiring a mixture of search, insertion, and/or deletion operations on collections of data, you will be able to compare the relative advantages of using arrays and linked lists in solving the problem efficiently.
4. Given a classical computational problem (e.g., infix-to-postfix conversion, postfix-expression evaluation, path planning, minimum-spanning tree computation), you will be able to trace a solution to the problem using appropriate data structures (e.g., stacks, queues, binary trees, binary search trees, red-black trees, graphs) and to predict the asymptotic run-time of the solution based on the selected data structures.
5. Given a collection of unordered data, you will be able to trace the execution of an advanced sorting algorithm (such as quick sort and heap sort) on this data set.
6. Given a set of data keys, you will be able to trace through a sequence of key insertions, searches and deletions on a balanced tree structure. you will also be able to discuss the relationship between the number of keys and the execution time of these operations.
7. Given a set of data keys, a hash function, a table size, and a collision-handling strategy, you will be able to trace through a sequence of key insertions and searches, and to discuss how varying the table size, hash function or collision-handling would affect the execution time of these operations.
8. Given a graph data structure, you will be able to implement it using either adjacency lists or an adjacency matrix, to traverse it using either a depth-first or breadth-first strategy, to identify its structural properties (whether it is directed, cyclic, connected, complete), and to trace the execution of one or more classical graph algorithms (e.g., Dijkstras shortest path, topological sort or minimum-spanning tree computation).
9. Given a problem requiring the efficient use of a variety of data structures, you will be able to apply object-oriented design principles in implementing and testing a solution to that problem in an appropriate object-oriented language.

Course Grading Policies

Your achieving these outcomes will come from a variety of activities, each of which will be factored into your grade based on the following weights:

Factor	Weight
Exams (3)	36%
Lab Exercises	20%
Programming Projects (5)	30%
Class Participation	7%
Completion of assigned exercises in textbook	7%

At the end of the term, your work in all of these areas will contribute to a numerical grade for the course based on a 100-point scale. Grade cutoff levels on this final scale are:

A $\geq$ 92	B $\geq$ 82	C $\geq$ 72	D $\geq$ 62
A- $\geq$ 90	B- $\geq$ 80	C- $\geq$ 70	D- $\geq$ 60
B+ $\geq$ 88	C+ $\geq$ 78	D+ $\geq$ 68	F $<$ 60

FAQ

Do I have to come to class? You are expected to arrive prepared to ALL the course sessions. Furthermore you are expected to participate in the classroom discussions and activities to the best of your abilities. Your doing this will not only ensure that you get the full 7% "class participation" component factored into your grade; more importantly the learning that transpires during these activities will help tremendously in preparing you for what is on the exams. It is difficult to envision a student missing and/or arriving unprepared to a number of the class sessions and still succeeding in the course.

What if I'm late in completing and submitting a lab exercise for evaluation? If you prepare beforehand, each lab is designed to allow you to finish within the one-hour period we spend in the lab on Monday. Should you not finish it in that hour, the lab exercise will have date prescribed date by which it must be finished. If it is not finished by that date, no credit is allowed.

What if I'm late in completing and submitting a programming project for evaluation? It will be accepted but will be penalized at the rate of 10% of point value the first day late, *an additional* 20% the second, *an additional* 30% the third ...

What if I'm late in completing the exercises assigned in the textbook? When an exercise is assigned, it will have a due date and time. If you complete it after that date and time, you will not receive credit for it. However, it would still be wise to understand how to complete it since many of these exercises will appear in slightly modified form on your exams.

If I miss an exam, can I make it up? If you are unable to take a scheduled exam, it may be possible to take a make-up exam provided that you do BOTH of the following, which are then subject to my approval:

- Make arrangements prior to the scheduled exam (for last minute emergencies, telephone me at 424-1388 or leave a message at the computer science office, 424-2068). No after-the-fact notifications will be accepted ... *AND*
- Have a written medical excuse signed by the attending physician OR have a note of justification from the Dean of Students Office.

Only one make-up exam will be given. It will be a rigorous comprehensive exam given at an arranged time during the last week of the semester.

Can I work with others on the lab exercises or on the programming projects? No, not in the sense of two people working together on the same problem. You may not "borrow" any piece of code or design of any length from someone else, unless you can live with a zero and the other potential academic sanctions of cheating (see UWO Student Discipline Code 2007, Chapter UW 14). However, it is acceptable to consult another student for help in debugging code that you have authored yourself and that is not producing the result you expected. If you do so, you must cite the help you received in the introductory documentation block of your program.