CS3610/5610D Data Structures

Spring 2023

Class Meetings:

Time: Tue, Thu, 9:30am - 10:50am
Room: Stocker 103

Recitation Sessions:

Time: Wed. 3:05-4pm
Room: Stocker 103

Instructor:

Dr. Jundong Liu
Office: Stocker 321A
Office Hours: Mon./Wed. 2-3:30pm, through TEAMs or in-person.
Email: liuj1@ohio.edu
Website: OU Blackboard

Teaching Assistant:

Mr. Ye Yue
Email: yy311820@ohio.edu
Office Hours: tentatively Mondays 3-5pm through TEAMs or in-person.

Prerequisites: (CS 3000 or MATH 3050) and C or better in CS 2401

Textbook:

Required Text: D. S. Malik, *Data Structures Using C++*, 2nd Edition, 2010.

Recommended Text: Mark A. Weiss, *Data Structures and Algorithm Analysis in C++*, 3rd Edition, 2007.

Grades and Exams

Written Homework, Quizzes, and Programming Projects: **50**% (Note: you need to earn a minimum of **20 points (40**%**)** out of this section to pass this course).

Exams: Midterm (**25**%); Final (**25**%) (Note: a minimum of **20 points (40**%**)** in this section is needed to pass this course.)

Final grades will be determined by applying the following scale. **90% and above = A, 80% and above = B, 70% and above = C, 60% and above = D, Below 60% = F**. This scale may be adjusted downward at the discretion of the instructor; however, it is very unlikely that the scale will be adjusted downward by more than 10%.

OU Undergraduate Catalog states that: "Once grades are submitted to the University Registrar, they are final and cannot be changed unless evidence of an error can be presented... Grades cannot be changed by arranging to complete additional work."

Important Dates:

Midterm Exam:	Mid March (tentatively).
Last day to drop:	March 31, Friday.
Final Exam:	Thursday, May 4, at 8:00 a.m.

Course and Attendance policies:

Regular class attendance is *strongly* recommended. Class attendance is not used in the final determination of grades. Students are required to attend class during the midterm exam and the final exam. **Makeup exams** will be given only to excused absences (defined under OU catalog), with documentable proof.

Homeworks and Academic Dishonesty:

The work you submit is expected to be your own. Plagiarism and other forms of academic dishonesty (e.g., **copying code or work from the internet, copying code or homework from other students, allowing other students to copy your code or work**, etc.) will be handled within the guidelines of the *Student Handbook*. Students who commit serious acts of academic misconduct **will receive an F in this class** and the University Judiciaries will be notified.

Projects assignments in this course were carefully designed by Dr. Liu and previous TAs. Please refrain from posting related materials on the internet.

Course Description:

The course emphasizes the importance of fundamental data structures and algorithms to programming. Here we build on CS2400/2401 and examine the following topics in more detail: (i) abstract data types, (ii) mathematical analysis of algorithms, (iii) trees and graphs, (iv) searching and sorting techniques, and (v) advanced data structures.

Course Outcomes

Outcome 1: Ability to analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions.

- An understanding and ability to use the basic terminology concerning asymptotic analysis.
- An understanding of the basic tree traversal techniques and their running times.
- An understanding of the basic operations on binary search trees and their running times.
- An understanding of the basic operations on graphs and their running times.
- An understanding of the basic data structures and algorithms associated with hash tables.
- An understanding of the basic data structures and operations on heaps and their implementations.
- An understanding of the basic graph data structures and terminology.
- An ability to analyze simple iterative and recursive functions.
- An understanding of the average and worst case analysis of the standard sorting techniques

Outcome 2: Ability to design, implement and evaluate a computing-based solution to meet a given set of computing requirements in the context of program's discipline.

- An ability to use tree traversal techniques for practical applications (e.g., evaluating expressions).
- An ability to use graphs and graph algorithms in programs to solve practical problems.
- An ability to solve simple summations and recurrence relations.