Course Syllabus

✎ Edit

MIAMI UNIVERSITY

CSE 274 - Data Abstraction and Data Structures

Course Information:

- **Instructor**: Dr. Suman Bhunia (Email: bhunias@miamioh.edu)
- **Office hours:** Wednesdays and Fridays, 4:30–5:30 p.m. at 380 McVey Data Science Building. If you can't make it to my office during these hours due to time conflicts with other classes, meet me after class or email me to schedule a meeting.
- **Class Interaction:**
  - Section A: Wednesdays and Fridays, 1:15 – 2:35 pm at Benton 002
  - Section B: Wednesdays and Fridays, 2:50 – 4:10 pm at Benton 002
- **Course Site**: Canvas
- **TA help sessions**:
  - Zoom Link: https://miamioh.zoom.us/j/84674297206?pwd=8UBTkEDVVmlUVeoFs4mLaE834nmyaR.1 (https://miamioh.zoom.us/j/84674297206?pwd=8UBTkEDVVmlUVeoFs4mLaE834nmyaR.1)

TA Name	Email	Time
Aneesha Valluru	vallurs@miamioh.edu	Sunday 4-6 p.m.
Ryan Garrity	garritrj@miamioh.edu	Monday 4-6 p.m.
Miles Clements	clemenmo@miamioh.edu	Tuesday 4-6 p.m.

- **Required Materials:**
  - **Textbook**: Data Structures and Abstractions with Java, 5th Edition By Frank M. Carrano and Timothy M. Henry
  - **Software**: Java SE Development Kit (JDK) - JDK, and Eclipse 2024-06.
    - It is your responsibility to make sure the code you write can be run using my version of Eclipse and Java, and the simplest way to do that is to use the same versions I am using. All code submitted must be compatible with JDK 17 or later. When you start a new project, you will be asked which version the project should be compatible with. 17 is safe. There are older and newer versions of Java. Choose 17 to be on the safe side. If your code doesn't work because you used some older version, there may be deductions in your grade.

Important dates:

Event	Date
Last Day to Drop Course with no Grade	Sept 13
Midterm Exam	October 2 (tentative)
Fall Break	Oct 11 – Oct 13
Last Day to Drop Course with 'W' Grade	Oct 28
Thanksgiving Break	Nov 27 – Dec 1
Final Exam	Finals Week

Course Description

Overview:

Abstract data types and their implementation as data structures using object-oriented programming. Use of object-oriented principles in the selection and analysis of various ADT implementations. Sequential and linked storage representations: lists, stacks, queues, and tables. Nonlinear data structures: trees

and graphs. Recursion, sorting, searching, and algorithm complexity.

Prerequisites:

CSE 271 with C- or above

Expectations:

Have fun. Always be programming. Always be solving problems. Come to class. Start homework right away. Turn homework in multiple times, on time. Ask questions. Help one another.

Course Outcomes:

1. **Select and use appropriate data structures, abstract data types, and algorithmic methods in problem solving**
    - Describe the purpose and semantics of abstract data types, including: matrices, lists, stacks, queues, sets, maps, trees, graphs, and priority queues
    - Describe the purpose and semantics of major data structures including: row-major order representation of matrices, 2D-array representation of matrices, array-based lists, linked lists, hash tables (both chaining and open addressing), binary search trees, adjacency lists, adjacency matrices, heaps, and graphs
    - Create programs that utilize the major data structures
    - Use appropriate data structures for solving problems
    - Describe the purpose and possible alternative implementations of common tree and graph algorithms including tree traversals (inorder, preorder, and postorder), depth first search, breadth first search, Dijkstra's algorithm, and topological sort
2. **Implement common data structures and algorithms**
    - Implement common data structures, including: row-major order and 2D-array representations of matrices, array-based lists, linked lists, hash tables (both chaining and open addressing), binary search trees, adjacency lists, adjacency matrices, and heaps
    - Implement common tree and graph algorithms including tree traversals (inorder, preorder, and postorder), depth-first search, breadth-first search, Dijkstra's algorithm, and topological sort
    - Combine multiple data structures to create an efficient solution to a problem (e.g. implementing an LRU cache using a list plus hash table, or efficiently changing the priority of an item in a heap in Dijkstra's algorithm using an auxiliary array)
    - Create or modify one's own classes to be used with library collection classes (e.g. create a class that can be used as a key in a hash table or binary search tree)
3. **Implement data structures and abstract data types using object-oriented programming principles**
    - Create data structures using polymorphism and inheritance
    - Create data structures using that use generics
    - Represent abstract data types using interfaces
    - Implement classes and data structures using standard Java interfaces and methods: equals(), Comparable, exceptions, hashCode(), and Iterator
    - Write JUnit tests to help identify and prevent errors.
4. **Determine time and space requirements of common data structure implementations and algorithms**
    - Describe the purpose of asymptotic notations (e.g., O) for algorithm analysis
    - Apply asymptotic notations (e.g., O) to analyze common data structure algorithms (insertion, removal, searching of array-based and link-based implementations), and sorting/searching algorithms
    - Describe the differences between worst-case running time, expected running time, and amortized running time and apply them appropriately.

Tentative Course Topics:

1. Object-oriented programming (1.0)
    - Implementing ADT interfaces
    - Generics
    - Inner classes
    - Exception handling
    - equals()
    - Implementing the Comparable interface
2. ADTs: bag, maps, set, list, stack, and queue (4.0)
    - Operations

- Array-based and linked-node-based implementations
- Traversal algorithms (iterative and recursive)
- Varieties of linked nodes (singly, circular, doubly)
- Application of standard Java collection libraries

3. Trees (3.0)
   - representations
   - traversal algorithms (iterative and recursive)
   - binary search trees
   - heaps
   - Self-balancing trees (e.g. B-trees, red-blue trees, AVL trees)
4. Testing (0.5)
   - Writing JUnit tests to ensure correctness of code
5. Tables (2.0)
   - Dictionaries
   - Hash maps
   - Applications
6. Graphs (1.5)
   - implementations: adjacency lists and adjacency matrices
   - algorithms: depth-first search, breadth-first search, Dijkstra's algorithm, topological sort
7. Algorithm complexity (1.0)
   - Complexity of ADT implementations
   - Space and time complexity of basic sorting/searching algorithms

Course Grading

Measures of Evaluation

Deliverable	Weightage
In-class work	5%
Quizzes	15%
Labs	15%
Homework projects	15%
Midterm Exam	25%
Final Exam	25%
Total	**100%**

In-class work: You will turn in code or a quiz or something on most days. We will use tools such as TopHat for submission. These need to be done in class. Turn in what you have at the end of class. If you are absent, you won't get credit for these assignments. I will drop the two lowest in-class work grades. I won't provide any makeup in-class assignments.

Quizzes: Online quizzes are given using the course website.

Labs: The content covered will be applied through guided lab/ project assignments.

Homework projects: Expect to do a lot of programming in this class. You will implement, use, and test a variety of data structures. An important part of the course is learning particular techniques to implement data structures and algorithms. Start each assignment as early as possible.

Exams: There will be one midterm exam and one final exam. All exams are cumulative, closed-book—**no make-ups for missed exams**. If you are absent on an exam, your grade for that exam will be zero.

Assignment Submission Policies:

- **Submission Method:** All assignments must be submitted through Canvas only. Submissions sent by e-mail and so on will not be accepted.

- **Late Submission:** One-day late submission will result in a 10% grade reduction; two-day late submissions will result in a 20% grade reduction. No submission will be accepted two days after the due date.
- **Always back up your electronic work!** Computer/network failures are a fact of life and are not justification for an extension. WRITE YOUR ANSWERS ALONE…learn to help one another without sharing any code.
- If you submit a scanned copy of a handwritten page, please scan it properly and make sure all the contents are readable.
- **Programming Style:** All programming that is submitted as part of a lab or a programming assignment should follow the departmental Programming Style Guidelines. Use your comments to help build understanding of what the code does. If you cannot comment on your code effectively, then it is likely that you do not fully understand what you are doing. In particular, study the formatting, commenting, and naming conventions. But most importantly, be consistent. Check Miami's Coding Guidelines here: https://miamioh.edu/cec/departments/computer-science-software-engineering/student-resources/coding-guidelines.html

Letter Grading Conversion:

Letter Grade	Range
A+	100%to97%
A	< 97%to94%
A-	< 94%to90%
B+	< 90%to87%
B	< 87%to84%
B-	< 84%to80%
C+	< 80%to77%
C	< 77%to74%
C-	< 74%to70%
D+	< 70%to67%
D	< 67%to64%
D-	< 64%to60%
F	< 60%to0%

Class Policies

Class Attendance Policy

Unexcused absences are not allowed in this course. In case of an absence, inform the instructor beforehand, if possible, and submit any work due on time. For more information, refer to Chapter 9 of the *Student Handbook*. Should a student become ill, it is the responsibility of the student to contact the instructor and keep the instructor appraised of the situation.

Taking notes

- You will sometimes be provided with electronic presentations to give you basic information. These are not a substitute for taking notes.
- Take notes during videos and activities.
- Lab activities often depend on using what you wrote in your notes.
- "Good notes" does not mean "Write everything". Be selective.
- Focus on writing sample code, diagrams, "notes to self".

Course Webpage & Communication

All course content (slides, videos, announcements, handouts, assignments, etc.) will be posted on the Canvas page for this course. We will use Canvas for all assignment submissions, as well as for the use of discussion boards, grading, and other means of communication. **You should ensure that your settings enable you to receive course announcements directly to your Miami email address so that you are immediately notified of any updates.**

Lecture Capture

Please be aware that classes might be recorded and shared only with students in this class. Students are not allowed to reshare the course lecture recordings.

Copyright Disclaimer

Course materials provided to you, including presentations, tests, outlines, and similar materials, are copyright protected by the faculty member(s) teaching this course. You may make copies of course materials solely for your **own use**. You may not copy, reproduce, or electronically transmit any course materials to any person or company for commercial or other purposes without the faculty member's express permission. Violation of this prohibition may subject the student to discipline/suspension/dismissal under Miami's Code of Student Conduct or Academic Integrity Policy.

Student Resources

Support for Students

As an instructor, I have a **duty to report** (https://www.miamioh.edu/policy-library/employees/general-employment/non-discrimination/duty-to-report.html) . This means I am required to promptly report to the Deputy Title IX Coordinator (**titleix@miamioh.edu** (mailto:titleix@miamioh.edu)) any information a student shares with me regarding harassment, discrimination, sexual misconduct and interpersonal violence, or retaliation. **A report does not initiate an investigation. It engages a discussion of your resources, supportive measures, and options available.** If students want to speak with someone confidentially, the following resources are available on and off campus:

- **Student Health Services** (https://gnldr.online/tracker/click?redirect=https%3A%2F%2Fmiamioh.edu%2Fstudent-life%2Fstudent-health-service%2Findex.html&dID=1578667353841&linkName=Student%20Health%20Services) , **(513) 529-3000** (https://www.google.com/search?q=Miami+University+Student+Health&rlz=1C1GCEB_enUS814US814&oq=Miami+University+Student+Health&aqs=chrome..69i57j46i39i175i199j0l3j69i60l2j69i618#)
- **Student Counseling Services** (https://gnldr.online/tracker/click?redirect=https%3A%2F%2Fmiamioh.edu%2Fstudent-life%2Fstudent-counseling-service%2F&dID=1578667353841&linkName=Student%20Counseling%20Services) , **(513) 529-4634** (https://www.google.com/search?rlz=1C1GCEB_enUS814US814&sxsrf=ALeKk02sACQS4DV0liKgWpTJGcUqvNjG1Q%3A1603376956794&ei=PJeRX8z8L4iPtAaSgpioDQ&q=Miami+University+StuECaABwAngAgAFbiAH_CplBAjE4mAEAoAEBqgEHZ3dzLXdpesgBCMABAQ&sclient=psy-ab&ved=0ahUKEwjMis3PtMjsAhWIB80KHRIBBtUQ4dUDCA0&uact=5#)
- Women Helping Women (WHW) Sexual and Interpersonal Violence Support Specialists are available to support all students and can be contacted by emailing mu@womenhelpingwomen.org. As well as calling/texting 513-846-8402 between 9AM-5PM. The 24-hour hotline is 513-381-5610. WHW supports ALL survivors of dating/domestic violence, sexual assault, and stalking, regardless of gender identity or sexual orientation.

Speaking with a confidential resource person does not preclude students from making a formal report to the University if and when they are ready.

https://miamioh.edu/diversity-inclusion/programs-resources/report-incident/index.html (https://miamioh.edu/diversity-inclusion/programs-resources/report-incident/index.html)

For more information, please visit **https://miamioh.edu/campus-safety/sexual-assault/** (https://miamioh.edu/campus-safety/sexual-assault/) and **https://www.miamioh.edu/diversity-inclusion/oeeo/index.html** (https://www.miamioh.edu/diversity-inclusion/oeeo/index.html) .

Accessibility Statement

Students with disabilities are encouraged to request reasonable accommodations. Student Disability Services (SDS) registration should be completed before the provision of accommodations. Please visit the Student Disability Services Website for more information. You can also contact SDS at 513-529-1541 or sds@miamioh.edu.
If you are eligible to receive accommodations, please schedule an office hours appointment at the beginning of the semester to discuss accommodation plans.

Mental Health Services

If you are a student experiencing mental or emotional distress, you are encouraged to call Student Counseling Service (513-529-4634). The Community and Counseling and Crisis Center (844-427-4747) has a 24-hour hotline for emergencies outside of business hours.

Academic Support

The following resources are available for you as a student:

- **Rinella Learning Center Academic Support.** (https://miamioh.edu/student-life/rinella-learning-center/academic-support/index.html)
- **Howe Center for Writing Excellence.** (http://miamioh.edu/hcwe/)
- **International Student Resources.** (https://miamioh.edu/academics/intl-student-resources/index.html)
- **Student Success Center.** (https://miamioh.edu/emss/offices/student-success-center/about/index.html)

Academic Integrity Information

Collaboration, and Academic Integrity Information:

The Department of Computer Science and Software Engineering is committed to maintaining strict standards of academic integrity. The department expects each student to understand and comply with the **University's Policy on Academic Integrity** (http://www.miamioh.edu/integrity/) the undergraduate student handbook and the graduate student handbook. Students may direct questions regarding academic integrity expectations to their instructor or to the department chair. All work submitted must be original for that class. Submitting the same project for two different classes is grounds for charging a student with academic misconduct unless prior written permission is received from both instructors.

"Problem Solving Assignments" are assignments that involve programming, math, proofs, derivations, and puzzles. The purpose of a problem-solving assignment is for you to develop the skills necessary to solve similar problems in the future. To learn to solve problems you must solve the problems and

write your solutions independently.

It is worth reiterating that the important aspect of the assignment is that you actually create the solution from start to finish; simply copying a solution and then understanding it after the fact is not a substitute for actually developing the solution.

The notion of academic integrity can be confusing in courses with substantial problem solving because certain forms of collaboration and investigation are permitted, but you are still required to complete your assignment independently. The following scenarios are meant to help distinguish between acceptable and unacceptable levels of collaboration and research, but are not all-inclusive:

ACCEPTABLE:

- Consulting solutions from code given to you by your instructor.
- Using code you wrote in this course this semester to help you write other code in this course this semester.
- Seeking help on how to use the programming environment such as the editor, the compiler, or other tools.
- Seeking help on how to fix a program syntax error or how a certain language feature works.
- Working through a problem on an at-home Canvas quiz collaboratively
- Discussing strategies with a fellow student on how to approach a particular problem. This discussion should not include significant sections of completed work or source code (including printouts, email, viewing on a monitor). Discussions should begin with a clean sheet of paper and end with conceptual drawings and/or pseudo-code.
- Searching Google or StackOverflow or similar sites for assistance with a non-central part of an assignment. For example, in an assignment about calculating the closest pair of points in a list, it would be OK to search for something like "how to loop through a list" or "how to add items to a list". However, you would then need to include a comment directly in the source code you used indicatint the origin of that code, with a working link to that code.
- Posting your code for projects you are proud of to your GitHub repository in order to help you when applying for jobs.

UNACCEPTABLE:

- Looking at another solution including those written by current students, past students, or outside sources such as code or solutions found on the Web, or in publications other than the current class textbook. In other words, looking at other solutions to a problem or a similar problem, even if it's just to "get ideas" about how to write your solution, is not permitted.
- Using code you wrote for a course in a previous semester.
- Giving or receiving some or all of the answers to an at-home Canvas quiz.
- Exchanging answers to an at-home Canvas quiz (example: I'll give you my answers to the first 5 problems, and you give me your answers to the next 5 problems).
- Using another solution as a starting point and then modifying the code or text as your own work.
- Collaboratively writing the source code for an assignment.
- Providing a copy of your solution or a portion of your solution, in any form (electronic, hard copy, allowing another student to view your code on a monitor), to another student.
- Giving or receiving code fragments to fix a problem in a program.
- Searching Google or StackOverflow or similar sites for assistance with a non-central part of an assignment. For example, in an assignment about calculating the closest pair of points in a list, it would not be OK to search for something like "code to find the closest pair of points".
- Posting your code to homework sharing sites such as Chegg.

If you are stuck on a problem and you are tempted to search for a solution on the Web or to look at another student's solution STOP and email or ask your instructor for help.

The default penalty for any instance of academic dishonesty in CSE will be a zero on the assignment followed by a reduction of a full letter grade in the course (for example, a C- would become a D-). This will be the case whether the judgment is reached in the Office of Academic Integrity or by the department chair.

Generative AI Tools are not permitted for code in CSE 274 - not for writing your code, and not for helping you think about code

Generative AI coding tools such as Copilot, CodeWhisperer, ChatGPT, and so on are great tools to assist us in certain situations. However, in CSE 274, where the goal of the course is to learn how to implement your own data structures and algorithms, generative AI tools get in the way of learning the material and are not permitted for any work in this course. This includes using generative AI to "get ideas" about how to write your code. If you need ideas about how to write your code, ask me. Note that the majority of this course is for your work on exams, which you will do on paper with no notes and no computers. There may be assignments that specifically ask you to use AI tools to assist you, but you must always begin with the assumption that those tools are not allowed.

However, you may use Generative AI Tools to help explain *concepts* to you as long as no code is generated. But it is your responsibility to verify its accuracy.

In other words, you may ask ChatGPT a question like "What are the pros and cons of linked lists compared to array lists? Note that if you get back false information, it's up to you to discover that. That is, if you get something wrong on an exam, you can't use "ChatGPT told me the wrong thing" as a defense.

Miami University Learning Community

Miami University is committed to fostering a supportive learning environment for all students irrespective of individual differences in gender, race, national origin, religion, handicapping condition, sexual preference, or age. Students should expect and help create a supportive learning environment free from all forms of prejudice. Disparaging comments, sexist or racist humor, or questioning the academic commitment of students based upon these individual differences undermines our learning community. If such behaviors occur in class, please seek the assistance of your instructor or department chair.