# CSCI 151
# Data Structures

Fall 2017
Lecture: MWF 8:10-9:00 (A1), MC Reynolds 315
Lab: T 1:10-4:00 (L7), WAC 249
Website: http://ozark.hendrix.edu/~yorgey/151/

Instructor: Dr. Brent Yorgey, MC Reynolds 310
Office hours: by appointment at
http://byorgey.youcanbook.me
Email: yorgey@hendrix.edu
**Note**: I generally do not respond to emails before 4pm.

## Course Description

Builds on skills acquired in CSCI 150, placing emphasis on object-oriented software design and data abstraction. Students are introduced to data structures (lists, stacks, queues, priority queues, trees, graphs, hash tables) and programming techniques such as recursion and sorting algorithms. Other topics covered include analysis of algorithm complexity and automated unit testing. Programming assignments focus on the design and implementation of algorithms and data structures using the Java language.

## Evaluation

Evaluation will be based on

- 25%: Weekly labs
- 35%: Projects
- 10%: Participation
- 30%: Two midterm exams

## Late days

Each student has four late days to spend throughout the semester as they wish. Simply inform me any time *prior* to the due date for an assignment (a lab or project) that you wish to use a late day; you may then turn in the assignment up to 24 hours late with no penalty. Multiple late days may be used on the same assignment. There are no partial late days; turning in an assignment 2 hours late or 20 hours late will both use 1 late day.

Late days are intended to cover all circumstances, both expected (*e.g.* you have three exams and a paper due at the same time) and unexpected (*e.g.* you get sick), so use them wisely. Additional late days will be granted only in exceptional circumstances.

## Labs

Much of your experience with programming in this course will be through weekly labs, which will comprise 25% of your final grade. Lab attendance is required. Each lab will be assigned at the start of the lab period on Tuesday at 1pm (though many may be posted earlier if you want to get a head start), with time allocated to work on the assignment during the scheduled lab period, and will typically be due in one week, before the start of the next lab.

On these labs, you may work with a partner on the lab assignments if you choose. Their name must be listed on any code you hand in as joint work. A partnership need only turn in a single copy of the assignment. If students working as partners wish to turn in a lab late, *both* students must use a late day.

## Projects

You will have four projects in this course, one about every 3 weeks, for a total of 35% of your final grade. These projects will cover concepts we have discussed in class and in labs, and will be due one to two weeks after they are assigned. In place of a final exam, you will complete an open-ended final project and present your project during the scheduled final exam time.

**You must complete the projects individually**. You may discuss concepts and ideas with your classmates, but the code you turn in must be your own. You will be graded not only on correctness, but also technique, documentation and evaluation of your solution. Further details on the grading standards and handin instructions for each project will be given when they are assigned.

## Exams

There will be two midterm oral exams: one the week of October 9–13, and one the week of November 13–17. The exams will consist of a mixture of short questions I will ask you on the spot, and more complex questions to which you will be able to prepare your answers ahead of time. During the week of the exam you will sign up for a time slot in which to come to my office to take the exam.

## Attendance and submission policies

Prompt lecture attendance is expected. Although I typically do not take formal attendance, unexcused absences may be reflected in your class participation grade. If you must be absent for some reason, please let me know in advance.

If you are absent from lecture (whether excused or unexcused) it is **your responsibility** to obtain notes from other student(s). Do not come to me and ask "what did I miss?".[1] On the other hand, if after obtaining notes you have specific questions or confusions regarding the topics covered, I would be happy to talk with you.

## Disabilities

It is the policy of Hendrix College to accommodate students with disabilities, pursuant to federal and state law. Students should contact Julie Brown in the Office of Academic Success (505.2954; `brownj@hendrix.edu`) to begin the accommodation process. Any student seeking accommodation in relation to a recognized disability should inform the instructor at the beginning of the course.

## Academic Integrity

All Hendrix students must abide by the College's Academic Integrity Policy as well as the College's Computer Policy, both of which are outlined in the Student Handbook.

For specific ways the Academic Integrity policy applies in this course, please refer to the Computer Science Academic Integrity Policy.

The short version is that academic integrity violations such as copying code from another student or the Internet are **easy to detect**, will be **taken very seriously**, and carry a default recommended sanction of **a zero on the assignment *in addition to* a decrease of one letter grade on your final grade**.

If you have any questions about how the Academic Integrity policy applies in a particular situation, please contact me.

[1] I am likely to answer that you missed the part where that is your responsibility.

## Learning objectives

By the end of the course you will be able to:

- Write and thoroughly test a medium-sized program (400 lines minimum)
- Implement and analyze the following data structures:
  - Linked lists
  - Stacks
  - Queues
  - Priority Queues
  - Hash tables
  - Trees
  - Graphs
- Implement and analyze different sorting algorithms
- State and apply a mathematical definition of algorithmic efficiency
- Articulate the difference between static and dynamic type systems
- Analyze the running time of an algorithm or data structure operation using "big-O" notation
- Understand and apply common complexity categories to select the right data structure and algorithm for a given task
- Use classes to implement abstract data types
- Use interfaces and inheritance to implement subtype polymorphism
- Use generics to implement parametric polymorphism
- Use and understand recursion as an implementation technique
- Employ automated unit testing to verify and document software functionality