*Note: Student work submitted as part of this course may be reviewed by Oxford College and Emory College faculty and staff for the purposes of improving instruction and enhancing Emory education.*

# Course Syllabus for
# CS 171 - Introduction to Computer Science II, Spring 2015

**Lecture, Integrated Lab Times**: MWF 1:15 – 2:20 PM, M 9:30 - 10:30 AM in Pierce 206

**Instructor**: Paul Oser
**Email**: poser3@emory.edu
**Office**: Pierce 122A
**Hours**: (CS 170 & 171 only) 12:00 – 1:15 PM MWF, 10:35 to 11:50 AM TTh and "Open-door" policy; (all Math and CS courses) 3:00 – 6:00 PM on MTuWTh

**Textbook**: Algorithms, Fourth Edition, Robert Sedgewick and Kevin Wayne

**Overview**:
This course is a continuation of CS170. Emphasis is on the use and implementation of data structures, and fundamental algorithms, with introductory algorithm analysis, and object oriented design and programming with Java.

Students will be given many opportunities to write programs to demonstrate their mastery of the algorithms and data structures covered in this course – especially via the lab assignments. Problem solving and real-world applications will play an important role, providing a driving motivation for developing and/or selecting appropriate algorithms or data structures to accomplish the associated goals as efficiently as possible.

**Goals for Student Learning**: Students at the conclusion of this course should be able to…

- Effectively use the following features of the Java language: polymorphism, interfaces, exceptions, and generic types
- Implement and use stacks, queues, linked lists, hash tables, and graphs and other abstract data types
- Do basic analysis of algorithms with regard to their running times
- Implement a variety of sorting algorithms, including selection sort, insertion sort, merge sort, quicksort, and heap sort
- Effectively design and implement recursive algorithms
- Implement and use basic algorithms from graph theory to accomplish various tasks

**Prerequisites**:
Successful completion of CS 170 or an equivalent course.

**How to approach the programming assignments in this class**:
Programming is a skill best learned by "doing" – so students should plan on spending a considerable amount of time in front of a computer outside of class this semester. Eclipse, the Java-related software on the laptops in Pierce 206, should be available on the computers in the Library and elsewhere on campus, if one not have access to a computer of their own (although most will likely find the latter much preferred).

Be aware that a certain amount of struggle in trying to figure out what's going on in a program, or why a particular error results is healthy – this is largely how programmers learn their craft. The labs are designed to do this – to put students into situations where they may need to experiment with several ways to proceed before settling on the one that seems to work best. In this light, students should avoid getting discouraged when things don't go smoothly the first time – or the first several times – they attempt a programming assignment. That said, when students are completely flummoxed – they are both encouraged and expected

to come talk to and get help from their instructor.

Java code can be compiled and run regardless of whether or not it follows the good naming conventions, or is properly indented, or has appropriate explanatory comments laced throughout the code, etc. Such programs may even produce correct output. That said, to be successful students should care a great deal about these things. To this end, all students are expected to follow the standard conventions of "good style" when programming, and points can and will be deducted when deviations from good style interfere with the grading of a student's program.

**Lab Assignments**:
The labs will constitute the bulk of the graded programming assignments for the course. Each lab may consist of one or more programs that students will need to submit. Lab assignments will involve designing, coding, testing and debugging non-trivial programs based on written assignment specifications. There will be several individual programming assignments and one or more team projects.

For the team projects, students should avoid the temptation to "divide and conquer" the assignment in question. Students giving into this temptation will only end up "doing" half of the work, and consequently, only really learning half of the material (which invariably gets revealed on test day). Allowing students to work with each other in teams (i.e., pairs) is intended to create conversations between students as they *both* develop each program in their own way -- conversations that will reveal when one approach is better than another, and should be adopted by both.

Students will be asked to periodically turn in their java programs via SFTP using an account provided to them on the math center server (i.e., mathcenter.oxford.emory.edu).

Students are required to regularly backup their workspace to protect against the loss and/or catastrophic failure of their computer.

Typically, all of the lab grades will be equally weighted. However, the instructor reserves the right to weigh some of these grades more heavily, so that if a particularly challenging program or lab is assigned, students can be rewarded for the extra effort involved.

**Exams**:
There will be three closed-book tests and a final exam. The three tests will emphasize reading/debugging code, conceptual understanding, and writing code. Doing well on each exam will strongly correlate to having read and understood the relevant sections of the text book, and having worked in earnest -- and successfully -- on the programs assigned up to that point in the class.

**Quizzes**: There will be some number of quizzes given throughout the semester over the readings in the text or content covered in class. They may or may not be announced ahead of time.

**Grading**:

| | |
|---|---|
| Individual Projects: | 30% |
| Team Projects: | 10% |
| Quizzes | 4% |
| Exams: | 36% |
| Final Exam: | 20% |

**Late Policy**:
Students are expected to be present for all scheduled tests. Any conflicts should be brought to the instructor's attention as soon as possible. If a legitimate reason exists for missing a test – as determined by the instructor – then the test must be taken prior to the regularly scheduled date. In the unusual circumstance where taking the test early is not possible, ***students should be aware that any make-up tests given will be designed to be more difficult to offset the additional time given for study***. Students must provide written documentation in advance of any special accommodations required for testing. This includes additional time or other needs. The final exam cannot be rescheduled.

In general, late programming assignments will not be accepted and missed quizzes may not be made up. This policy will be waived only in an "emergency" situation with appropriate documentation, and at the instructor's discretion.

**Honor Code Policy**:
All class work is governed by the Oxford College Honor Code. No collaboration is allowed on tests or exams. Unless otherwise indicated by the instructor, individual programming assignments must be completed without consulting code written by other students or resources outside of those provided. That said, students are welcome to discuss general principles and concepts about the assignments with each other and with the instructor. For the team projects, students will work in groups of two or three.
For team projects – ALL students' names that worked as a team must appear at the top of every file submitted for that team project. These names should be formatted as a java comment, similar to the example below:

```
/*******************************************/
/**** Lab Partners: Mike Beck & Jon Fowler ****/
/*******************************************/
```

Students in a team may not turn in a single project between them – each person should turn in his or her own project – even if it is substantially similar to that of the other members of their team (which in most cases would be expected). Collaboration on team projects between students that are not on the same team is not allowed, and does a serious disservice to all involved on test day*. Really.

** This means that if the instructor starts to get the impression that inappropriate collaboration on labs is occurring, which nullifies the capability of the labs to provide a genuine measure of students' ability to write code, the instructor will be forced to ask more questions on the tests that will require students to write code "on the spot". As such code-writing is done with pencil or pen, and without the benefits of code-completion, alerts to syntax errors, or a debugger - like those the Eclipse IDE provides - students will undoubtedly find such tests more difficult.*