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, Fall 2017

Lecture, Integrated Lab Times: MWF 9:30 – 10:35 AM, Tu 1:40 – 2:40 PM in the Old Dining Hall Classroom

Instructor: Paul Oser
Email: poser3@emory.edu
Office: The front modular building between the Chapel and Williams Hall
Hours: MWF 10:35 – 11:35 AM, 2:20 – 3:45 PM, TuTh 2:40 – 3:45 PM along with an "open-door" policy;

Web: http://mathcenter.oxford.emory.edu/cs171
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: inner classes, interfaces, exceptions, and generic types
- Implement and use various abstract data types, including (but not limited to) stacks, queues, linked lists, binary search trees, heaps, hash tables, and graphs to accomplish various tasks
- Implement a backtracking search algorithm
- Analyze the efficiency of algorithms with regard to how their running times increase as the related problem size increases, in both the average and worst-case scenarios
- Implement a variety of sorting algorithms, including selection sort, insertion sort, merge sort, quicksort, and heap sort, understanding the advantages and disadvantages of each
- Implement and use data structures and algorithms related to graph theory to accomplish various tasks

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

"Administrivia":
Java and the Eclipse IDE, the software primarily used in this course, should be available on the computers in the Library and elsewhere on campus. That said, students should have access to a laptop that they can bring to class, and use to work on their labs. In the (hopefully unusual) circumstance where a student does not have access to such a computer, he or she should let the instructor know as soon as possible, so that whatever options might exist can be explored and pursued in a timely manner.

Students will be asked to periodically upload java programs via SFTP to the math center server (at

mathcenter.oxford.emory.edu), and then officially "turn them in" via an SSH connection with this same server. The programs graded may include programs that were partially fleshed out in class, or programs left to the student to complete solely on their own.

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

Students should check the Canvas site for this class **daily**. All announcements and assignments for this class will be posted there.

Notes to supplement or clarify the textbook readings, sample code, lab assignments, and review exercises will be able to be found online at http://mathcenter.oxford.emory.edu/cs171/. The collection of links at this website will grow over the course of the semester and the schedule listed there may be modified as the semester progresses, so students should check this site daily as well.

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.

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 just 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 and/or consistent 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. In a given lab period, one or more lab assignments (i.e., programs) may be assigned. The time given to complete these assignments will minimally be one week, although it may often be longer. These lab assignments will involve designing, coding, testing and debugging non-trivial programs based on some written specifications.

There will be several individual programming assignments and one or more team projects.

The lab periods are designed to give students time to work on the lab assignments in a focused way, and seek advice of the instructor as necessary. When a lab period is used for a team project, it also gives one valuable time to collaborate with his or her partner(s). Unless a student has already submitted the final versions of all of the programs assigned for a given lab period, he or she is expected to be present in class for the entire lab period and working to these ends. Note: failing to take advantage of this time for these purposes can affect one's lab grades.

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.

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 conceptual understanding, and reading/writing/debugging code. Doing well on each exam will strongly correlate to having read and understood the notes online, 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:	7%
Quizzes	4%
Exams:	39%
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; this policy will be waived only in an "emergency" situation at the instructor's discretion, and only after any documentation the instructor requires has been provided.

Honor Code Policy:
All work done in this class is governed by the Oxford College Honor Code.

Students may not give, access, or receive any information not expressly permitted by the instructor on tests or exams. Collaboration between students on tests and exams is strictly prohibited.

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 can work in groups of two or three.

If two students work together, a comment similar to the following (with the two names changed appropriately, but everything else absolutely identical) must be included as the first line of every .java file submitted for the project:

```
//SUBMITTED BY: Mike Beck & Jon Fowler
```

If three students work together, the comment should take the following form (with the three names changed appropriately, but everything else absolutely identical):

```
//SUBMITTED BY: Mike Beck & Jon Fowler & Velina Veleva
```

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.*

Special Accommodations:
Access, Disability Services and Resources (ADSR) works with students who have disabilities to provide reasonable accommodations. In order to receive consideration for reasonable accommodations, students must contact ADSR and complete the registration process. Faculty may not provide disability accommodations until an accommodation letter has been processed; accommodations are not retroactive. Students registered with ADSR who receive a letter outlining specific academic accommodations are strongly encouraged to coordinate a meeting time with their professor to discuss a protocol to implement the accommodations as needed throughout the semester. This meeting should occur as early in the semester as possible. Contact Access, Disability Services and Resources for more information at (770) 784-4690 or adsroxford@emory.edu. Additional information is available at the ADSR website at

http://equityandinclusion.emory.edu/access/students/index.html