

*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 170Q - *Introduction to Computer Science***

### **Fall 2014: Section 09A/L-B (Course# 8803 & 8804)**

**Lecture and Integrated Lab:** MWF 9:30 – 10:35 PM & Tu 1:40 – 2:40 PM in Pierce 206

**Instructor:** Paul Oser **Email:** [poser3@emory.edu](mailto:poser3@emory.edu) **Office:** Pierce 120B **Hours:** (CS 170 only) 10:35 to 11:35 AM on MWF and “Open-door” policy; (all Math and CS courses) 3:00 – 6:00 PM on MTWTh

**Web:** <http://www.oxfordmathcenter.com/drupal7/node/10>

**Textbook:** None required. All notes and coding examples will be available online at the website above.

**Other Required Materials:** a Dropbox account, which can be obtained for free -- see <http://www.dropbox.com>.

**Overview:** This course is an introduction to computer science for the student who expects to make serious use of the computer in course work or research. Topics include: fundamental computing concepts, general programming principles, and the Java programming language. Emphasis will be on algorithm development with examples highlighting topics in data structures.

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

- Effectively use primitive data types and common pre-made objects in the Java language
- Effectively use program-flow-control concepts (i.e., "for"-loops, "while"-loops, "if"-statements, etc.)
- Effectively use arrays and strings for storing and manipulating a large amount of data
- Build classes and objects of their own design
- Effectively use subclasses and interfaces to facilitate Object-oriented design
- Begin to become familiar with event-driven programming

#### **Prerequisites:**

There are no official prerequisites although some familiarity with email and web browsers will be helpful. Knowledge of high school algebra and basic problem solving skills are assumed. This course is the first of a two-semester sequence for computer science majors and is followed by CS 171.

#### **The “Ways of Inquiry” at Oxford:**

“Ways of Inquiry” courses are designed to introduce students to the specific ways knowledge is pursued in each discipline through active engagement in the discipline's methods of analysis. INQ courses start with questions, are student-centered and often collaborative, and they place increasing responsibility on students for their own learning. Students not only experience each discipline's distinctiveness but also move beyond its boundaries to understand connections with other disciplines and fields.

#### **The “Ways of Inquiry” Used in this Course:**

Writing a computer program is an act of inquiry. There is no "recipe" that can be given to students so charged. On the contrary (and in a quite literal sense) – with every program they write, students "create their own recipe" to accomplish the task in question.

Students will be given many opportunities to write programs – especially through the lab assignments. They will have a goal (a task their program must perform); they will have the tools they need (the language specification and API); and how they get to that goal is up to them. Students will attack these tasks in a variety of ways, some elegant, some brutish, but all will have to pull up their sleeves and get down in the trenches of figuring out how this new thing can be done with what they know.

The instructor's job in this course is to demonstrate the requirements and capabilities of the Java language, give students some good guiding principles, and then largely get out of their way – letting them discover how to use this language to do what they want to do. The instructor will also play a supporting role: 1) helping students see how certain "fundamental" questions can guide their efforts in accomplishing the goals given to them; 2) driving students away from inefficient solutions; and 3) revealing to students (through questioning) cases they haven't considered, that might cause their program to behave in a manner contrary to what they intended – and thus alerting them to the need to debug as necessary.

Students will have the opportunity to pair up with other students as they engage in many of their programming tasks – and the tasks themselves will often connect in some way to interesting problems or subfields of mathematics, cryptography, and/or other disciplines.

### **Lab Work / Assignments / Blackboard:**

*Students should plan on spending a considerable amount of time in front of a computer outside of class this semester.* Students should make every effort to work all of the programs assigned, as programming is a skill best learned by “doing”. The Java-related software on the laptops in Pierce 206 should also be available to you on the specialty computers in the Library and in the Kaleidoscope Lab, should you not have access to a computer of your own.

Assignments will involve designing, coding, testing and debugging programs based on a written assignment specification. These programs will involve a conceptual understanding of language features and require skill with various software tools. With programming it is important to “work smarter, not harder.” Brute-force approaches often lead to long, tedious, unsuccessful hours of work. The right approach can help you write correct, easy-to-understand and efficient code with minimal effort. Programs should be completed individually or as otherwise directed, although you are welcome to discuss general principles and concepts about the Java language with other students and the instructor.

Students will be asked to periodically turn in java programs via an SFTP upload to the mathcenter server at [mathcenter.oxford.emory.edu](http://mathcenter.oxford.emory.edu) (or possibly via a shared folder on their Dropbox account). 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 their Blackboard account for this class **daily**. All announcements and assignments for this class will be posted there.

### **Exams:**

There will be three closed-book tests and a final exam that will test student understanding of the material. The three tests will each be comprised of two sections: one emphasizing reading and debugging code, and one emphasizing writing of code. Doing well on the first part of each exam will strongly correlate to having read and understood the notes online and other reference material provided. Doing well on the second part of each exam will strongly correlate to having worked in earnest - and successfully -- on the programs assigned up to that point in the class.

### **Final Project:**

Each student will code (possibly with a partner) a final project of their own design (in consultation with the instructor) to demonstrate their mastery of the language and methods covered in this course.

## Grading:

Lab Assignments: 25%

Exams: 45%

Final Project: 10%

Final Exam: 20%

Each lab assignment will be given two grades:

One of these grades will be -- of course -- for your programs producing the correct output or desired behavior.

The second grade given for each lab will be for “good style and comments”. Typically, one program from those assigned for a given lab will be selected (pseudo-randomly) and checked to see if

- 1) Appropriate comments laced throughout the code that clarify, explain, and/or otherwise document the details of that program and how it works (60%)
- 2) Standard coding conventions have been followed (*e.g., proper indentation throughout your code; the selection of good variable names; the appropriate use of constants*) (30%)
- 3) The code is efficient in both its writing and execution (10%)

The instructor reserves the right to weigh some labs or individual programs more heavily, so that if a particularly challenging program or lab is assigned, students can be rewarded for the extra effort involved.

## 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 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. For the labs and final project, however, students may work in pairs if they wish. IMPORTANT: if two students do work as a pair on any given lab – BOTH students’ names must appear at the top of every file submitted for that lab. So that this extra text doesn’t cause an error during compilation, it should be formatted as a java comment, similar to the example below:

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

Lab partners may not turn in a single lab between them – each person should turn in his or her own lab assignment – even if it’s substantially similar to their partner’s (which in most cases would be expected).

Collaboration on lab assignments between students that are not lab partners is not allowed, painfully obvious to spot by the instructor, and does a serious disservice to all involved on test day. Really.