Eric Roberts and Mark Hopkins
CSCI 121
Handout #1
August 27, 2018

CSCI 121 — Course Information

Lectures: Professor Eric Roberts

E-mail: **esroberts@reed.edu**
Office: Library 316
Phone: 503-517-4883
Hours: Tuesdays, 9:00 A.M.–12:00 NOON
Fridays, 11: 00 A.M. –1:00 P.M.

Labs: Professor Mark Hopkins

E-mail: **hopkinsm@reed.edu**
Office: Library 314
Phone: 503-517-4735
Hours: Tuesdays, 4:45–6:00 P.M
Wednesdays, 10:00–11:00 A.M
Thursdays, 10:00–11:30 A.M

Course description

CSCI 121: Computer Science Fundamentals I
Full course for one semester. An introduction to computer science, covering topics including elementary algorithms and data structures, functional and procedural abstraction, data abstraction, object orientation, logic, and the digital representations of numbers. Emphasis is on mathematical problems and calculations and on recursive algorithms and data structures. The course includes a significant programming laboratory component where students will solve computational problems using a high-level language. The mechanisms for processing and executing programs will be surveyed. Prerequisite: three years of high school mathematics. Lecture-laboratory.

Lectures

Lectures are scheduled for Monday, Wednesday, and Friday at 3:10 P.M. in PSYCH 105. A tentative lecture schedule for the semester appears on the last two pages of this handout.

In addition to lecture, you must also sign up for a weekly 80-minute laboratory, led by Mark Hopkins.

Web page

The web page for CSCI 121 is **https://www.reed.edu/~esroberts/csci121**. All the materials and course announcements will be posted on this website.

Course materials

The text for this class is a set of course notes that Eric is developing this semester. In addition, we will also distribute additional material in the form of class handouts.

Examinations

CSCI 121 will hold an in-class midterm examination on Wednesday, October 3, and a final examination as scheduled by the registrar.

Assignments and projects

This semester, CSCI 121 requires weekly programming assignments that consist of short programming exercises or problems for you to work through by hand. Those assignments will ordinarily be distributed each Monday and due on Friday. In general, you will be expected to complete some of the problems during the Tuesday lab sessions and then finish the others on your own schedule. In addition, there will be five larger programming projects for which the due dates will be given on the project handouts. The programming projects will be graded on the following scale:

++ An absolutely fantastic submission of the sort that will only come along a few times during the semester, if at all.

+ A submission that exceeds our expectations. The program must reflect additional work beyond the requirements or get the job done in a particularly elegant way.

✓+ A submission that satisfies all the requirements for the assignment—a job well done.

✓ A submission that meets the requirements for the assignment, but which is either stylistically weak or has a few minor problems.

✓– A submission that falls significantly short of the requirements for the assignment.

– A submission with still more serious problems but that nonetheless shows some effort and understanding.

–– A submission that shows little effort and does not represent passing work.

Due dates and late days

Assignments and programming projects must be submitted electronically as described in the first assignment handout. All assignments are due at 11:59 P.M. on the date indicated on the assignment handout. Assignments submitted after 11:59 will be considered late.

Because each of you will probably come upon some time during the semester where so much work piles up that you need a little extra time, every student begins the semester with two free "late days," where "day" is defined as a calendar day. Thus, if your assignment was due on Friday but turned in on Saturday, that assignment would be one day late. After your late days for the semester are exhausted, programs are assessed a late penalty of one category point per late day used (a ✓+ turns into a ✓, and so forth). Late days are valuable, and it pays to keep some around for the harder assignments toward the end of the term.

In special circumstances such as extended medical problems or other unforeseeable emergencies, extensions may be granted beyond the late days. To request an extension, send email to either Eric or Mark no later than 24 hours before the program is due.

Contests

In addition to the projects, we will also schedule three programming contests at different points during the term. The point of these contests is to give you a chance to show creativity and initiative beyond what is formally required by the course. Rules for each contest will be distributed in class when they are announced.

To encourage greater participation in the contests, we offer two additional incentives. First, every reasonably serious entry gets you one virtual ticket in a random drawing for a special grand prize at the end of the semester. The more contests you enter, the more chances you have. Winning runner-up prizes or honorable mentions in a contest give you additional chances. The random drawing will take place at the beginning of the review section for the final exam.

This semester, you have an additional opportunity to win virtual tickets. The draft reader is being written as the semester proceeds and will certainly contain errors. If you find a bug in the reader, send it to **esroberts@reed.edu**. If you're the first person to report a problem that turns out to be real, you get a ticket for the random drawing.

Grading

The most important component of the course in terms of learning is certainly the programming assignments and projects. To give you a strong incentive to complete them, it is important that assignments count for a significant proportion of the grade. At the same time, exams are necessary both to see whether you have retained mastery of the material and as a defense against the possibility that some of you will get too much help on the assignments, either appropriately from the teaching assistants or inappropriately by copying code from other students.

At most universities and colleges, computer science courses have had more than their fair share of Honor Code cases. Since neither of us has previously taught at Reed, we do not yet have a good sense as to whether to what extent cheating is likely to be a problem here. As we go through the semester, we will keep a close eye on the assignments to get a sense of the severity of the problem. If we don't uncover a significant amount of cheating, we'll weight the assignments more heavily in the calculation of the final grade. If we do, we'll count the exams for more.

Tentative lecture schedule

Monday	Wednesday	Friday
August 27 Course overview Introducing Python Programming by example	29 Data and types Variables and values Arithmetic expressions	31 Strings and Boolean data Control statements Lists and iteration
September 3 Labor Day (no class)	5 The Portable Graphics Library Simple graphics	7 Functions Arguments and parameters The mechanics of functions
10 Graphics revisited Decomposition Top-down design	12 Libraries and interfaces The random library	14 First-class functions Event-driven programming Responding to mouse events
17 Defining simple classes Encapsulating private data Timer-based animation	19 The `GObject` hierarchy More `GObject` subclasses	21 Binary representation Representing characters
24 A simple machine model References and addresses	26 Strings and characters Common string patterns	28 String applications
October 1 Debugging strategies	3 Midterm exam	5 Cryptography The Enigma machine
8 Recursive functions Recursion and efficiency Thinking recursively	10 Lists revisited Operations on lists	12 Stacks and queues

Monday	Wednesday	Friday
October 22 Searching algorithms Simple sorting algorithms	24 Algorithms and efficiency Better sorting algorithms	26 Computational complexity Big-*O* and Ω notation
29 Multidimensional arrays Pixel arrays	31 Image manipulation	November 2 Working with files
November 5 Dictionaries and maps Hashing	7 Classes revisited Standard class methods	9 Overview of Adventure!
12 Principles of OOP Inheritance	14 Designing class hierarchies	16 Linked lists
19 Linked structures Trees and graphs	21 Amazing algorithms	23 Thanksgiving break (no class)
26 Immutable types Tuples Python's type system	28 Iterators and generators List comprehension	30 Exception handling
December 3 Encapsulation in Python Closures can be closed	5 Frontiers of computing "Oh, the places you'll go"	