

Rhodes College Digital Archives - DLynx

CS 360-01, Programming Languages, Fall 2009

Item Type	Text
Authors	Breck, Eric
Publisher	Memphis, Tenn. : Rhodes College
Rights	Rhodes College owns the rights to the archival digital objects in this collection. Objects are made available for educational use only and may not be used for any non-educational or commercial purpose. Approved educational uses include private research and scholarship, teaching, and student projects. For additional information please contact archives@rhodes.edu . Fees may apply.
Download date	2026-03-05 08:32:57
Link to Item	http://hdl.handle.net/10267/15726

Syllabus **Programming Languages**
CS 360 Fall 2009
CRN: 10358

Time Monday, Wednesday, Friday 1-1:50PM
Location BARRET 033 (computer lab in basement of library)
Instructor Eric Breck (<http://faculty.rhodes.edu/brecke>)
Office 419 Ohlendorf (901-843-3725)
Office Hours Office Hours will be posted on my website; please send e-mail to make an appointment if you cannot make office hours.
Email brecke@rhodes.edu
To ensure a quick response, the subject line of your emails should read
 cs360: [subject of question]
Do not assume I will read and reply to your messages instantaneously.

Moodle

All assignment submission will be via moodle. Do not e-mail me assignments. Course announcements and any updates to this syllabus will be posted via moodle. Hardcopy submissions for assignments that are not programs are acceptable, though dispreferred.

Book

The course textbook is *Programming Languages: Principles and Paradigms*, by Allen Tucker and Robert Noonan. Supplemental material may be distributed in class.

Prerequisites

Computer Science 241 (i.e., three semesters of programming), or permission of instructor.

Course Description

CS360 introduces fundamental programming language concepts, presenting design issues of the various language constructs, and examining the design choices for these constructs in a range of the most popular contemporary programming languages. Topics covered (not necessarily in this order) will include:

- Syntax of programming languages (T&N, Chapter 2)
- Issues in naming (T&N, Chapter 4)
- Types and type systems (T&N, Chapter 5)
- Semantics (T&N, Chapter 7)
- Functions (T&N, Chapter 9)
- Memory management (T&N, Chapter 11)
- Imperative programming (T&N, Chapter 12)
- Object-oriented programming (T&N, Chapter 13)
- Functional programming (T&N, Chapter 14)
- Logic programming (T&N, Chapter 15)

Exams

There will be weekly quizzes, two midterms and one final exam.

- **Midterms:** Friday, 2 October and Monday 16 November, in class
- **Final Exam:** Tuesday 15 December, 8:30AM
- Make-up exams will only be given in extreme circumstances.

Grade Breakdown

- 50% Assignments (there will about ten assignments).
- 15% Midterms
- 25% Final
- 5% Class Participation
- 5% Quizzes

Course policies

The instructor reserves the right to alter this syllabus as necessary.

Assignment submission

Assignments will generally be distributed on moodle immediately following the due date of the prior assignment.

The first line of each program source code file submitted for credit must be a comment that states your name and the assignment number. Assignments should be submitted via Moodle (by class time on the due date). Assignments received after class time on the date due are considered late. You are allowed to use the course textbook and the course notes for these programs. The use of any other material is forbidden.

Any assignments that require the submission of something other than program source code must be submitted as either plain ASCII text or Adobe PDF - no Word documents, please. To create PDFs on a Mac, choose Print from any application, then Save as PDF. To create PDFs on a PC, choose Print from any application, then choose PDFCreator as your printer of choice, and follow the prompts – this software is available on all campus computers, and can be downloaded free for your own machine. There are also scanners in the campus labs that can be used to create a PDF from a handwritten document. For **assignments that are not program source code only**, hardcopy submissions are also acceptable, though dispreferred.

Attendance

Attendance is expected at each class lecture period. You are responsible for learning any material presented during a class that you miss. Lecture notes and slides and textbook chapter references may be offered, but are not intended as a substitute for attending lecture – additional material may be presented beyond what is covered in these references.

Backups

Each student is responsible for keeping a back-up copy on disk of all files turned in for an assignment. Failure to do so could result in loss of credit for an assignment.

Classroom behavior

Please respect me and your classmates and do not browse the web, check e-mail, send or read text messages, or have private conversations during class.

Collaboration

You are expected to work individually on assignments. However, you are allowed and encouraged to discuss high-level details of the assignments. If group work is allowed, it will be mentioned explicitly in the assignment. Do not share your own code or copy others' code. Tools such as MOSS make it easy for me to check if you have copied code and just changed the whitespace and variable names; don't do it.

Communication with me

Like any human, my memory is imperfect. Therefore, if you make a verbal, in-person request or suggestion during class or elsewhere, you should follow up with an e-mail containing the same information. This includes but is not limited to: scheduling meetings, asking for regrades or other grade changes, indicating errors on the moodle site or in assignments, and anything else that requires me to remember to do something later.

Grade calculation

Grades for exams and assignments appearing in moodle are the recorded grades for each item. Aggregate grades appearing in moodle should be taken only as an estimate of the final grade. The final course grade will be calculated as specified above. Let me know as soon as possible if you believe a grade was recorded incorrectly or if you are unclear about how grades will be calculated.

Grading programs

Programs will generally be graded using the following guidelines.

- A (100 pts): The program is carefully designed, efficiently implemented, well documented, and produces clearly formatted, correct output.
- A- (94 pts): This is an ‘A’ program with one or two of the minor problems described for grade ‘B’.
- B (88 pts): A ‘B’ program could easily have been an ‘A’ program, but it may have minor/careless problems such as poor, inadequate, or incomplete documentation; several literal values where symbolic constants would have been appropriate; wrong file names (these will be specified per program assignment); sloppy source code format; minor efficiency problems; etc. (This is not an exhaustive list.) You would be wise to consider ‘B’ the default grade for a working program — this might encourage you to review and polish your first working draft of an assignment to produce a more professional quality final version of your program.
- C (75 pts): A ‘C’ program has more serious problems: incorrect output for important special cases (the “empty” case, the “maxed-out” case, etc.), failure to carefully follow design and implementation requirements spelled out in the assignment description, very poor or inefficient design or implementation, near complete absence of documentation, etc.
- D (60 pts): A ‘D’ program compiles, links, and runs, but it produces clearly incorrect output for typical cases. Or, it may deviate greatly from the design or implementation requirements stated in the assignment description.
- F (35 pts): Typically, an ‘F’ program produces no correct output, or it may not even compile. It may “look like a program” when printed as a hard copy, but there remains much work to be done for it to be a correct, working program.

Late assignments

Late assignments will be accepted, with a penalty of one letter grade per day. If a genuine emergency situation arises, please see me and we will work something out.

Network availability

You are required to take an exam at the specified time regardless of network accessibility. In other words, keep a copy of the practice tests, notes, etc. on **your** computer or network folder in case you have issues logging onto the network.

Numeric to letter grade conversion

Grades during the course will generally be given as numbers. The final course letter grade will be determined from the aggregate numerical grade using the following scale:

A	:	[93%, 100%]
A-	:	[90%, 93%]
B+	:	[87%, 90%]
B	:	[83%, 87%]
B-	:	[80%, 83%]
C+	:	[77%, 80%]
C	:	[73%, 77%]
C -	:	[70%, 73%]
D	:	[65%, 70%]
D-	:	[60%, 65%]
F	:	[0%, 60%]

Office Hours

Please come to office hours! You are not bothering me, I am here to help. During the office hours posted on my website, I will be in my office and available to answer your questions, help you with assignments, etc. You do not need to e-mail me to let me know you are coming. If for some reason I will be unavailable, I will try to send a notification via e-mail.

If you cannot make any posted office hours because of schedule conflicts, let me know by the end of the first week of class what your availability is, and I'll try to add another one.

You are also welcome to make an appointment, which consists of you sending me an e-mail saying "Can I come by at x time?", me responding "yes", and you showing up. Please do not miss such appointments.

Finally, you can also just stop by my office and see if my door is open. It often is, but won't always be.

Programming style

Programming is not a dry mechanical process but an art form. Well-written code has an aesthetic appeal while poor form can make other programmers and instructors cringe. Programming assignments will be graded based on correctness and style. Good programming practices usually include many of the following principles:

- Concise comments that summarize major sections of your code
Comments should be correctly spelled and grammatical.
- Meaningful variable and function names
- Well organized code
- White space or comments to improve legibility
- Avoidance of large blocks of copy-and-pasted code

Scholastic Behavior and Academic Integrity

Plagiarism, cheating, and similar anti-intellectual behavior are serious violations of academic ethics and will be correspondingly penalized. If you are concerned about a possible violation of this kind, please talk with me. I understand the pressure that students may experience while at Rhodes, and I will try to help as best as I can.

All projects and tests must be the student's own work, unless otherwise instructed by your instructor. Copying all or part of a project assignment, or downloading code or text from the Internet and submitting it as your own, or having someone else write code or text for your assignment, or having someone else debug your assignment, or allowing someone else to copy from you, or writing or coding or debugging someone else's assignment — these are all included in the definition of reportable Honor Code violations for this course. If you have any doubts about whether or not a work practice on a project assignment is acceptable, please clear it with the instructor before proceeding.

Special Accommodation

If you are in need of special accommodations, please register with the Office of Student Disability Services (<http://www.rhodes.edu/disability>) as soon as possible so that all necessary arrangements can be made. Accommodations cannot be made unless you register; they also cannot be made retroactively.

Workload

It is important to stay current with the material. You should be prepared to devote regular weekly hours to this course. More specifically, you should devote at least 2-3 hours for each in class lecture. You should expect to spend significant amount of preparation for this course working on a computer to try out example programs and to develop the programming assignments. **Do not wait until the last minute to start your programming assignments.**