

EECS 476 -Programming Language Design

Instructor: Reed

University of Illinois at Chicago -Fall '97

Class: Mon., Wed., Fri. 11:00 - 11:50 in LH 313

Office: SEO 917

Phone: 413-9478

email: reed@eecs.uic.edu, World Wide Web: <http://www.eecs.uic.edu/~reed/>

Office Hours: 3:15 - 4:30 M W F

Or by appointment on M, W, or F mornings. I will not be available on T or Th

Prerequisites: EECS 370 - or - MCS 261 & MCS 270

Text:

Sebesta, Robert W. Concepts of Programming Languages, 3rd. ed.
Addison-Wesley, 1996. ISBN 0-8053-7133-8

Grading

40% ~7 quizzes and/or homeworks (probably including 1 or 2 short programs)
30% Midterm: Wed., Oct. 8
30% Final (comprehensive), Thurs., Dec. 11, 8:00 - 10:00

100% Total. Final grades are curved at the end of the semester.

Note that the homeworks will probably include 1 or 2 short programming assignments. Assigned programs are due by 10:00 A.M. on the day due. Turning in a program consists of turning in your code using the UNIX *turnin* program. The time stamp from *turnin* will be used to determine the submission time. You must also turn in a printed copy of your program at the beginning of class on the day programs are due. Failure to turn in a paper copy will result in a 10 point penalty. Do not modify your program after it has been turned in. In case of a "turnin" problem, the last modification date of your original program can still be verified. If you want to change it, make a copy first. *The lowest quiz/homework/program grade will be dropped* at the end of the semester.

Please write the last two digits of your social security number (or some other number that you will remember!) on the upper-right hand corner of everything you turn in to me (e.g. programs, quizzes, tests). This helps me put papers in order to make it easier to hand them back to you.

You will be given the opportunity to take a make-up exam only in cases of medical or personal emergencies, which must be verified. If such an emergency occurs, call me or leave a note (or phone message) with the department secretary as soon as possible. If you will be out of town when an exam is scheduled, I *must* be told in advance and may require you to take the exam early. Otherwise, if you miss an exam you will receive **0** points.

Class attendance is not mandatory, however you are responsible for all information (handouts, announcements, notes, etc.) covered during class. You should ask fellow classmates for missed information, not the instructor or the T.A.

No incompletes will be given for poor performance in the course. No “extra work” or extra credit will be given.

If you feel that you deserve more points than you have been given on a quiz, assignment, or test, you must see the instructor about this *within one week* of the time the work in question is first returned to the class. After this deadline, no claims will be considered, justifiable or not.

Outline

We will mostly go straight through the book as noted below. We will omit Ch. 15 (Object-Oriented Programming Languages), as there is a separate course on that.

Ch. 1 - Background to studying Programming Languages (PL's)

Ch. 14 - Logic PL's (PROLOG)

Ch. 2 - Evolution of PL's

Ch. 3 - Syntax & Semantics

 Lambda Calculus

Ch. 4 - Names, Bindings, Type Checking, Scope

Ch. 5 - Data Types

 Midterm Exam (Wed., Oct. 8)

Ch. 6 - Expressions & Assignment Statement

Ch. 7 - Control Structures

Ch. 8 - SubPrograms

Ch. 9 - Implementing Subprograms

Ch. 10 - Abstract Data Types (*time permitting*)

Ch. 11 - Concurrency (*time permitting*)

Ch. 12 - Exception Handling (*time permitting*)

Ch. 13 - Functional PL's (LISP)

References

Pratt, Terrence W., and Zelkowitz, Marvin V. Programming Languages: Design and Implementation, 3rd ed. Prentice Hall, 1996. ISBN 0-13-678012-1

Fischer, Alice E., & Grodzinsky, Frances. The Anatomy Of Programming Languages. Prentice Hall, 1993, ISBN 0-13-035155-5.

Sethi, Ravi. Programming Languages: Concepts and Constructs, 2nd ed. Addison Wesley, 1996. ISBN 0-201-59065-4.

MacLennan, Bruce J. Principles of Programming Languages: Design, Evaluation, and Implementation, 2nd ed. Oxford University Press, 1987. ISBN 0-19-510583-4

Cheating:

Any student caught cheating on an exam or program will automatically fail the course and may be referred to the department chair and/or dean.

When writing programs, you may consult with *me* or the *TA* at any stage of your program development. Bring a current print-out, as well as a current copy on disk. You may only consult *others* **after** you have attempted to run your written program. Such consultation is limited to brief (less than 15 minutes) discussions with computer advisors, students, and others. You may seek help about the system or the editor from anyone at any time.

To avoid cheating via collaboration, do not show any other classmates your code. If a classmate consults you for help after attempting to run his or her program, briefly (less than 15 minutes) assist in determining why his or her code doesn't work, but refrain from suggesting specific new code. Do not lead your classmates into temptation: guard your print-outs.

Further Information, Handouts, Programs, Hints:

General course information will be distributed via my home page on the World Wide Web (<http://www.eecs.uic.edu/~reed/>).

Programs Grading Criteria:

If a program doesn't compile and run on the Sun workstations in SEL 2260, it will not be accepted and will receive a grade of **0**. Partial credit will be given on programs that do compile and run. Each program will be graded out of **100 points** as follows:

55% Runs correctly: conforms to assignment description for input and output, follows instructions given. Make sure to *test* your program thoroughly.

45% Programming style, further broken down as follows:

10% Meaningful Variable Names. Variable names should indicate their purpose. Names should be words connected by underscores or separated using capitalization, such as *gradesSum* or *grades_Sum*. Short loops of ~ 5 lines can use loop counter variables such as *i* or *j*.

10% Comments. Every function must have a short description stating the purpose of the function, what it receives, and what it returns. Comments should be easily identifiable. I should be able to understand your program by reading *only* the comments. Also include a header at the top of your program (see example below).

10% Functional Decomposition. A segment of code that appears more than once should be extracted to form a separate function. Functions should be no longer than around 50 lines.

10% Appropriate data and control structures: Global variables should be avoided and used only when necessary. Function parameters should be used instead. Appropriate looping and decision structures used.

5% Code Layout. Different nested levels should have different indentation, where statements at the same level should have the same indentation. Indent at least **3** spaces. Use either spaces or tabs consistently

100 points total

Remember - you will lose 10 points if you neglect to turn in a print-out.

Each program should include a descriptive header at the top of the first page such as the following:

```
/* -----  
  Description: Solve world hunger and bring a lasting peace.  
                (Come on, this is a graduate class. Did you expect something simpler?)  
  Author: Dale Reed  
  Date: 1/1/97  
  Class: Assignment # 1 for EECS 476  
  System: CC compiler on ernie.eecs.uic.edu  
  Input: Advice from all mothers who have lost children ina war or to famine  
  Output: 5-year action plan  
  -----  
*/
```