Fall 2008

CS 480/680: Comparative Programming Languages

Michael L. Raymer
Wright State University - Main Campus, michael.raymer@wright.edu

CS 480/680 – COMPARATIVE PROGRAMMING LANGUAGES
FALL, 2008

Instructor: Dr. Michael Raymer
391 Joshi
775-5110
michael.raymer@wright.edu

Room & Time: Mon & Wed, 4:10 – 5:25 pm, 248 Oelman Hall

Office Hours: Mon/Wed 2:30 – 4:00 pm, *or by appointment.*

Textbooks: D. Thomas and A. Hunt, *Programming Ruby* (Electronic) [TH]
D. Sitaram, *Teach Yourself Scheme in Fixnum Days* (Electronic) [DS]

Lecture Schedule:

Week	Topics	Reading
1	Compiled and Interpreted Languages	Slides
2	Language Specification	Slides
3	Functional Languages & Scheme	DS 1, 2, 3
4	Scheme Parameters, Variables & Scope	DS 4, 5, 6
5	Scheme Macros, Structures & I/O	DS 7, 8, 9
–	**Midterm Exam** *(Tentatively Wed. 10/8)*	—
6	Object Oriented Languages & Ruby	TH 1, 2
7	Containers, Blocks, Iterators	TH 3, 4
8	Inheritance, Mixins, Regular Expressions	TH 5, 6
9	Exceptions, I/O; Writing an Interpreter	TH 7, 8
10	A Scheme Interpreter in Ruby	—
–	**Final Exam: Mon. 11/17, 5:45-7:45pm**	—

Grading: Midterm Exam = 25%
Final Exam = 30%
Lab and homework assignments (~8) = 45%

90 – 100 = A; 80 – 89.9 = B;
70 – 79.9 = C; 60 – 69.9 = D;
< 60 = F
I *may* curve the final letter grades based on the overall distribution of scores.

Web Page: http://wisdom.wright.edu – Check this page often for announcements, assignments, and other important information. (The course archives can be found at http://birg.cs.wright.edu/cs480)

See the "Course Policies" handout for important information on the acceptable coding standards for the programming assignments, academic integrity policies, and other course policies.

CS 480/680 – Comparative Programming Languages
Course Policies

I. Late Assignments

PROGRAMMING ASSIGNMENTS (LABS) are due by 11:55pm on the due date. Late programming assignments will be accepted, but 10% of the total available points will be deducted for each day late. Labs are considered one day late after 11:55pm on the due date. At 11:55pm of each successive day (including weekends) the lab is considered an additional day late until turned in. Once a graded programming assignment has been returned, the assignment will no longer be accepted.

OTHER ASSIGNMENTS are due at the beginning of class on the assigned due date. Late homework assignments will not be accepted.

II. Academic Integrity

Discussion of course contents with other students is an important part of the academic process and is encouraged. However, it is expected that course programming assignments, homework assignments, and other course assignments will be completed **on an individual basis**. Students may discuss general concepts with one another, but may not, under any circumstances, work together on the actual implementation of any course assignment. If you work with other students on "general concepts" be certain to **acknowledge the collaboration** and its extent in the assignment. Unacknowledged collaboration will be considered dishonest. "Code sharing" (including code from previous quarters) is strictly disallowed. "Copying" or significant collaboration on any graded assignments will be considered a violation of the university guidelines for academic honesty. If the same work is turned in by two or more students, **all parties involved will be held equally accountable for violation of academic integrity**. You are responsible for ensuring that other students do not have access to your work: do not give another student access to your account, do not leave printouts in the recycling bin, pick up your printouts promptly, do not leave your workstation unattended, etc. If you suspect that your work has been compromised notify me immediately. NOTE: Failure to attend the first day of class, during which time I will explain these academic honesty policies in detail, does not excuse you from following these policies. If you have any questions about collaboration or any other issues related to academic integrity, please see me immediately for clarification.

III. Coding Standards

Program Orgnization

General format

The format that every file should follow is:

(a) *block comment including your name*
(b) preprocessor statements, constant definitions, etc.
(c) function definitions (if any)
(d) main program

NOTE: Not all items will be included in every file. EVERY file should include a block comment with your name. The rest of the items should be used in the order given when needed.

Use small functions

Keep functions small. The code for a function should be limited to no more than what will fit on the screen. About 50 lines should be considered an upper bound.

Comments

Use a block comment at the beginning of each file

Each source code file should have a comment block starting on the first line of the file giving the name of the file, the name of its creator, a description of the contents, a description of known bugs, restrictions or limitations and other general information about the file. Any additional information required for the assignment should be included in this comment.

Use a block comment before each function

Each function should be preceded by a block comment describing the function, its parameters (and the data type for each parameter), its return value (and type), side effects (such as the use of [shudder!] global variables) and any known bugs or limitations.

Use inline comments

Inline comments can be helpful in definitions but should be used sparingly in procedural code. *Every major section of code needs an inline comment for clarification.*

INDENTATION AND SPACING

Indent consistently using a reasonable style

Each control construct (function definition, loop, if, switch...) should define a new indentation level.

Don't be chintzy with indentation

Use at least 3 spaces for each indentation level.

Use whitespace to improve readability

Use white space to separate variables, operators, and other syntactic units.

Example:

```
x=2*x+y; <-- hard to read
x = 2 * x + y; <-- easier to read
```

Also add one or more blank lines to separate major sections of code.

NAMING CONVENTIONS

Use readable English names

Constants, variables and functions should be given names that are good meaningful English. Keep names like I, J and Z to a minimum. When names are stated in the program assignment handout, they MUST be used.

Use abbreviations sparingly

A few common abbreviations (like Numb for Number) are common enough that they won't be misunderstood. It is a good idea to comment variables so their meanings are clear. If a variable is redefined, an explanatory comment is essential.

GENERAL

Only necessary files should be turned in: no executables, old programs or labs.

Any special information should be in a file called README.TXT.

Miscellaneous

At this level, all programs should compile/interpret correctly, and have been well tested. There should be no infinite loops, system locking, or output that doesn't make sense. It is your responsibility to seek help for syntax and logic errors or unreasonable output. The programs should run according to the handout requirements.

If there is a problem with any of these, a README.TXT file should be included that contains all pertinent details of the problem, what steps were taken to correct the problem and any ideas you have as to what is wrong.