WELCOME TO
Computer Science 342
PRINCIPLES OF PROGRAMMING LANGUAGES

<table>
<thead>
<tr>
<th>Professor</th>
<th>Hridesh Rajan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Office</td>
<td>101 Atanasoff Hall</td>
</tr>
<tr>
<td>Phone</td>
<td>294-6168</td>
</tr>
<tr>
<td>Email</td>
<td><a href="mailto:hridesh@cs.iastate.edu">hridesh@cs.iastate.edu</a> or <a href="mailto:cs342s@cs.iastate.edu">cs342s@cs.iastate.edu</a></td>
</tr>
<tr>
<td>Office Hours</td>
<td>Tuesday 3:30 - 5:00pm</td>
</tr>
<tr>
<td>TA</td>
<td>Sean Mooney</td>
</tr>
<tr>
<td>Office</td>
<td>0145 Pearson Hall</td>
</tr>
<tr>
<td>Email</td>
<td><a href="mailto:smooney@cs.iastate.edu">smooney@cs.iastate.edu</a></td>
</tr>
<tr>
<td>TA Office Hours</td>
<td>Tuesday and Thursday 1:00 - 2:00pm</td>
</tr>
</tbody>
</table>

1 INITIAL INSTRUCTIONS

- In the course web page, go to "Lecture Notes" for lecture outlines
- Read the forward and the preface of the recommended textbook “Structure and Interpretation of Computer Programs” (SICP). This book is available on-line.
- Start reading the Little Schemer, chapters 1-4
2 PROGRAMMING LANGUAGE

Definition 2.1 A programming language is a language that is capable of expressing all computations.

Exercise 2.1 What are other programming languages?

Definition 2.2 A general-purpose language is one that is not tailored to some particular application area.

Examples: AspectJ, Eos, Ruby, Python, Java, Ada, PL/I, BASIC, COBOL, Pascal, Ptolemy, SNOBOL, Icon, Algol W, Algol 60, Algol 68, C, Haskell, Edison, ... (literally thousands)

But the techniques we’ll study aren’t just limited to general-purpose programming languages.

Definition 2.3 A special-purpose language is a language that is designed to support some particular application.

These don’t have to be able to compute all computable functions.

Examples: Lotus 1-2-3 macros, the MFC class library, MUMPS, JavaScript, VBA, Cold Fusion

Exercise 2.2 What are other formal languages?

Formal languages are also used in a variety of settings: music, knitting, mathematics, engineering, ...

3 THE COURSE IN A NUTSHELL

- The journey to the mountain:
  - functional programming techniques
- The mountain climb:
  - using functional techniques to write interpreters
- Our equipment:
  - Scheme, a Lisp dialect
- Our guides:
  - the textbook authors

4 Com S 342 ESSENTIAL OBJECTIVES

You should be able to:

1. Write and modify programs in functional style
2. Make effective use of data abstraction
3. Change or enhance interpreters to have features such as:
(a) control flow,
(b) variables,
(c) recursion,
(d) scoping,
(e) syntactic sugars,
(f) arrays,
(g) parameter passing mechanisms,
(h) type checking,
(i) objects, and inheritance

4. Write programs using such features and explain their behavior
5. Explain the data structures and algorithms used in interpreters.
6. Compare alternatives in design and implementation.

these are the essential ideas in PLs.

4.1 What is functional style?
Use functions as arguments etc. to help modularize your code

4.2 Why these objectives?
- help you think about good ways to program, abstraction mechanisms
- (see below for functional style)
- data abstraction is key to making programs easily modifiable
- learn new languages quickly mapping ideas from this class to the language
- read/write language reference manual
- help you program more effectively
- most programs evolve into languages, or have language parts

5 REASONS FOR USING FUNCTIONAL STYLE
- Makes certain programs clearer
  - can see all inputs eval(exp, env)
- functions as arguments helps in:
  - abstraction
  - modularization
- gives you more ways to solve problems
- ideas important for:
– writing concurrent programs
– algorithm design
– specification
– describing programming languages
– web services

• it’s interesting and fun

6 SKILLS WE TEACH IN 342

• functional abstraction techniques
• data abstraction
• how to build interpreters
• meta-linguistic abstraction

7 PREREQUISITES

We intend to enforce the following formal prerequisites as best we can, if you don’t have these, drop this course.

• Com S 321
• Engl 104
• Com S 330 or Cpr E 310
• Com S 309 or Com S 361 or Com S 363

For example, consider the following ideas that will be helpful during the course of this semester.

Q: In what course do you learn about the halting problem?
Q: Context-free grammars? Data abstraction?
Q: Machine representation of data? The run-time stack?
Q: Recursion?

8 ESTIMATED TIME NEEDED FOR HOMEWORK

Think of Com S 342 as like a math course. There are frequent problems to do and we go fast. The following is a rough estimate of necessary efforts.

• Average: about 12 hrs/week.
• Mode: about 13.5 hrs/week.
• Range: 0-5 to more than 14 hrs/week.
9 TEXTBOOKS

9.1 Required

- Essentials of Programming Languages (3rd edition) [3]
- The Little Schemer (4th edition) [2]

9.2 Recommended

- Structure and Interpretation of Computer Programs (2nd edition) [1]

Also there is stuff on reserve and on the web

10 GRADING

- No curve grading
- Your grade is
  - 70% based on tests
  - 30% on homework

  Moral: use homeworks to learn, cheating on homework won’t help

Other policies such as those for late homework, exams, etc, are described on the course web page

Exercise 10.3 In groups of 3 (or 4) for 5 minutes:

1. introduce yourselves
2. write down at least 1 question about the policies, course specification, or syllabus that you agree is confusing or you want to talk about
3. select someone to ask the question(s)
This quiz is not worth any points but solely for your review and to help you realize when you don’t understand something from class. Following answers could be either true or false.

1. This course is graded on an absolute grading scale (i.e. there will be no curve).

2. Extra credit points are applied at the end of the course to my total homework score to determine my final grade.

3. If I do poorly on the exercises, I will only lose the points for exercises - it will not be considered subjectively in the final grade (beyond its 5% contribution).

4. If I do poorly on the homeworks, I will only lose the points for homeworks - it will not be considered subjectively in the final grade (beyond its 30% contribution).

5. Instructional staff can meet with you outside of office hours by appointment.

6. This course has an online form for anonymous feedback.

7. Lecture attendance is required. Recitation attendance is not.

8. Lecture outlines are posted on the course web site.

9. If you cannot understand what the professor is saying, the proper response is to just be quiet and look it up later because everyone else understands.

10. The main reason to learn Scheme (even if you hate it) is because it’s used everywhere in business so you won’t be able to avoid it.

Following are short answer questions.

11. The e-mail address to write to for quickest reply from the instructional staff is:

12. You should check your department Unix account at least how often for course information?

13. According to Prof. Rajan, the number one thing to remember about Scheme is that:

References

