CSC151 Functional Problem Solving

Fall 2012

Synopsis: This class introduces "computational thinking" as a method of solving problems. Using visual media (art and images) as a vehicle for exploration, you will explore important concepts from computer science that can be applied to a variety of fields. With daily labs that help you become progressively fluent in a high-level programming language, you will practice the design and analysis of algorithms, recognizing and creating abstractions, and using recursion. You also get to create some fun pictures, to boot.

Instructor: Jerod Weinman
Office: Noyce 3825
Phone: x9812
E-mail: [weinman]

Mentor:
E-mail:

Course web page: http://www.cs.grinnell.edu/~weinman/courses/CSC151/2012F/
Class meetings: 11:00-11:50 a.m., MTWF, Science 3813

Contents

1 Accommodations

If you have any disability that requires accommodations, please meet with me right away so that we can work together to find accommodations that meet your learning needs. You will also need to provide documentation of your disability to the Dean for Student Academic Support and Advising, Joyce Stern, located on the 3rd floor of the Rosenfield Center (x3702).

Please also note that I require your accommodations. The chemical fragrances found in lotions, after shave, body sprays, scented laundry products, perfume, cologne, etc. make many people who suffer with asthma, allergies, environmental sensitivities, cancer, and migraines much sicker. I am sensitive to many chemicals you may not even notice, so please try to avoid using such scented products before coming to class and *especially* if you visit my office.

2 Overview

The official catalog description reads:

> A lab-based introduction to basic ideas of computer science, including recursion, abstraction, state, information-hiding, and the design and analysis of algorithms. Includes introductory programming in a high-level, functional language. Prerequisites: None.

This means that you'll be introduced to the basics of computation, learning how to adequately describe and decompose problems of a computational nature so that you can effectively tell a computer the steps it should take to solve the problem. We will study some beginning concepts that make this process possible, easier to undertake, and often elegant.

Our major objectives for this course include:

- Understanding some *fundamentals* of computer *science*: algorithms, data structures, and abstraction.
- Familiarity with the *practice* of computer *programming* (design, documentation, development, testing, and debugging) in a high-level programming language, Scheme.
- Learning problem-solving from a functional programming perspective.
- Sharpening *general* thinking and learning skills.

Why take it? Some basic knowledge about solving problems computationally can be useful in careers involving ***every*** major and concentration offered at Grinnell. (I invite you to challenge me with one if you are skeptical.) This course will outfit you with some versatile concepts that are applicable to most other programming languages. Plus, you get to be creative in many different ways, from problem solutions to works of art. Creativity is highly encouraged!

Important warnings

Important! Experience shows that CSC 151 exercises different parts of your brain than other courses (even math and science courses). Expect some challenges, but have *confidence* that you can work through them and that you'll come out of the course with much more knowledge.

Important! Like learning a foreign language, learning in this course is cumulative: New ideas often build on ideas from earlier in the course. If you feel like you've missed something important, please come talk to me right away.

Important! Computers have no common sense or compassion. They are complex, and sometimes they do things we don't expect. *When things go wrong,* ***don't blame yourself***. Ask me or a tutor for help.

In addition, we are using software (MediaScript) that has been developed at Grinnell and may still have some bugs lurking within. It is entirely possible that it may crash for no good reason. Thus, you should develop a habit of saving your work frequently (or subconsciously twitch control-S to save, as I do).

3 Textbooks / references

There is no required textbook for this course, as the material has been written by Grinnell faculty and will be posted on the web. However, there are some useful textbooks and references you may want to bookmark or keep handy.

Books

Dybvig, Kent R., *The Scheme Programming Language* (Third Edition), MIT Press, 2003.
`http://www.scheme.com/tspl3`
An effective reference and guide to learning the Scheme language.

Felleisen, Matthias, Findler, Robert Bruce, Flatt, Matthew and Krishnamurthi, Shriram, *How to Design Programs*, MIT Press 2001.
`http://www.htdp.org`
A wonderful guide, particularly for later topics in the course like higher-order programming, re-factoring, and recursion.

References

Kelsey, Richard, Clinger, William, and Rees, Jonathan, editors, *Revised*^{5} *Report on the Algorithmic Language Scheme.* 1998.
`http://www.cs.grinnell.edu/~weinman/courses/CSC151/doc/r5rs.pdf`
The complete, concise definition of the Scheme programming language version we are using.

4 Schedule of topics

The following is an approximate schedule of topics to be discussed during the course. See the web page schedule for details.

Week	*Topic*	*Week*	*Topic*
1	Introduction	8	Geometric art
2	Numbers, procedures, graphics	9	Characters, vectors, data structures
3	Design, images, color	10	Analysis, testing, association lists
4	Documentation and iteration	11	Higher-order procedures, search
5	Booleans and conditionals	12	Sorting
6	Lists and recursion	13	Objects
7	Preconditions, local procedures	14	Files, summary, review

5 Assignments and activities

This class is taught in a collaborative workshop style. Some days we will spend working through problems or concepts in an interactive fashion. Most days, you'll work on laboratory problems on the computer with other students.

To make class time most valuable for you, I do not plan to lecture on material that is covered in the reading. Instead, because experimenting and practicing is the best way to learn, you'll have the opportunity to ask questions and then begin working collaboratively on the day's lab exercises with the instructor and a class mentor available to provide assistance.

By studying the day's topics beforehand, we can concentrate the beginning of class on areas of confusion. Because our class time is limited, you must come prepared to each class, meaning you should:

1. Check the schedule for the day's meeting to find out the topic.
2. Study the assigned material *before* class.
3. Come to class on time, with paper and something to write with, ready to *participate*.

Under a normal 16 credit load, I expect that you will spend *at least* (but hopefully more) 40 hours per week on your studies (class time, homework, and studying).[1] Thus, you should plan to spend a *minimum* of 10 hours/week on work for this course:

5.1 Reading

Nearly every day there will be a brief reading assigned. Before each class, you should check the course web page for the reading. Reading the material may entail the following:

Overview You should look over the reading once to get an overview of the material to be covered (see section headers and summaries).

In-depth Next, study the material closely. You should try some of the examples in the text as you read. Perhaps not everything will make sense at this point, but hopefully many or most things will.

Final notes After carefully reading the material, make a few notes to yourself about what you think are the most important concepts being covered, as well as any questions you have.

These important readings are short, but may introduce non-trivial ideas. Some details will make more sense as you try things out on your own in lab, but I am always happy to take questions, which may be answered or deferred to a hands-on exercise.

[1] This is a minimum recommendation for achieving "satisfactory" (i.e., C-level) results. "Good" or "excellent" results may require a greater investment.

5.2 Participation

Because much of our work in this course involves collaboration and discussion, you will be evaluated on your participation.

Participating in class involves:

- being present in class (physically and mentally)
- coming to class on time
- coming to class prepared
- asking questions when appropriate
- making positive contributions to class discussion by volunteering and when called upon
- staying on task during lab exercises, and
- working effectively with your lab partner(s)

Students who regularly meet these criteria can expect to earn 90% for their participation grade. I will reward students who regularly provide significant insights or guide discussion in productive ways with a higher participation score. Students who fail to participate regularly (e.g., demonstrating a lack of preparation or involvement during lab exercises) or who participate in counterproductive ways (e.g., by dominating the conversation, making inappropriate comments, or getting off-task) can expect to earn a lower score.

5.3 Labs

Most class days will involve collaborative laboratory work. You might not complete the laboratory assignments during class. It is very important that you finish these outside of class to be sure that you are engaging in all the covered material. Like playing an instrument or speaking a foreign language, the only way to become proficient is to practice, practice, practice! You will be expected to carefully write up your solutions to some lab exercises. Lab write-ups will be announced at the end of class. For this reason, but most importantly for your records, you should keep a careful record of your work as you progress through the lab.

Deadline When a lab is to be written up, I will let you know at the end of the class session in which it is conducted. Lab write-ups are due by the beginning of class on the specified date. You can view which labs are due and when on the course web labs page.

Submission You and your lab partner(s) may submit your write-ups jointly (if you complete it together) or individually (if you complete it separately, outside of class).

If you complete the lab on your own, acknowledge your partner's contributions in your writeup. If you submit the lab together, only one submission is required. However, all group members are jointly responsible for ensuring a submission is made. (For example, if your partner told you he would submit a lab but he forgets to, you will both receive a zero.) For this reason, I recommend submitting labs while you are sitting together and including all named group members in the Cc: line of the e-mail.

Submit your lab write-up in the body of an email (do not send an attachment) by the beginning of class the day it is due. That means the top of the hour sharp, not two minutes after, when it will be considered late.

The subject of the email should be of the form "CSC 151-02 Lab: *Title of Lab*".

Format Your write-up should include your solution to each exercise in the lab. If the exercise directs you to write Scheme code, include your Scheme code. If the exercise directs you to find out what the output of some expression is, copy and paste the output from the MediaScript console. If the exercise asks a question or asks you to explain something, write an answer in English. You do not need to copy the entire lab into your write-up (please don't!), but do include the problem number for each exercise.

Grading Lab write-ups will be graded on a binary scale, 0 or 1. You will earn a 1 if your write-up includes a solution or evidence of serious effort for each exercise in the lab, and a 0 if the write-up was not turned in or if some exercises are not attempted. If you were not able to complete some exercises because of their difficulty, you can still earn full credit by explaining where you got stuck and coming to talk with me as soon as possible.

In short, every diligent student should earn full credit for lab write-ups.

5.4 Homework

A homework assignment is due each Tuesday night at 11:59 PM (when there is no project or exam). The homework is intended to let you learn and apply some new algorithms and ideas, rather than merely checking that you've learned the basic concepts (as the quizzes do). However, they are **not intended to be huge undertakings**. If you find the assignments are consistently taking longer than 3-4 hours, please talk to me.

I will give you instructions about what forms of collaboration are permitted on each homework assignment. Since I want to give you room to go above and beyond the requirements of the assignment, homework will be graded on a simple ternary scale:

PLUS	(105%)	Exhibits exceptional insight, creativity, and/or craftsmanship
CHECK	(90%)	Meets the requirements of the assignment.
MINUS	(70%)	Does not meet the requirements of the assignment.

I expect most work to receive checks. Of course, no credit will be given if nothing is submitted. The course schedule includes eight homework assignments; your lowest homework grade will be dropped.

5.5 Quizzes

Each Friday, there will be a short, written quiz covering one or two key ideas from the previous week. Because lab work is done in pairs, the quizzes are intended to check each individual student's understanding of these ideas in a timely fashion (i.e., well before the examinations). If you (or the class as a whole) are missing a key concept, I want to revisit that concept as soon as possible so we can build on it in later lessons. Moreover, studies show that testing is a surprisingly effective learning device.

The course schedule includes 12 quizzes; your lowest quiz grade will be dropped. Because the goal of the quizzes is to check that you have learned basic skills, an answer that is basically correct will receive full credit, even if there are minor syntax issues. For code, my rule of thumb is that the code is basically correct if I believe you could quickly make it work at the computer (and it uses the required ideas). A partially correct answer will receive partial credit at my discretion.

If you require accommodations for these quizzes, please come talk with me as soon as possible.

5.6 Project

The course will also include one more substantial project, in which you will have an opportunity to exercise your creativity in working on a program over two weeks. I will inform you of the grading rubric when the project is assigned.

Project Assigned	Wednesday 79 November
Detailed Proposal	Tuesday 13 November
Complete Submission	Tuesday 20 November

5.7 Exams

As opportunities for you to demonstrate your programming and design prowess, there will be three take-home exams and an *optional* final exam. The following are their due dates (11:59 PM), though they will be distributed 5-7 days in advance of the due date.

Exam 1	Tuesday 25 September
Exam 2	Tuesday 6 November
Exam 3	Tuesday 4 December
Final Exam	Thursday 20 December (2 pm)

You should find the exams challenge you to go beyond what we have done in class.

Midterm take-home exams Take-home exams are *open notes, open computer, and open instructor*. However, because I intend the exams to assess your own individual understanding of the material, **collaboration on exams is not permitted**. You should not talk to anyone (except me) about take-home exams before they are due. You should not discuss the problems on the exam, nor your answers, with anyone else. You should not give any information about your progress on the exam (e.g., which problems you have completed or which you found difficult), and neither should you ask others about their progress. If you have any doubt about what is and is not permissible, ask the instructor.

Final exam The final exam can be used to replace your lowest score on one other examination. Although the final for this class is optional, you may end up deciding to take it. Therefore I encourage you not to make airline reservations that will conflict with your final exam schedule.

Unlike the others, the final will be an in-class exam. Talking with other students during the exam will not be permitted. You may ask me questions. The exam will be closed-book and closed-computer, but you will be allowed to use one double-sided, 8.5"x11" sheet of hand-written notes.

6 Grading

My goal is for everyone taking this course to be able to demonstrate familiarity, fluency, and excellence with the course concepts. I would be very happy if you all met the goals above and received “A”s. The following weighting of individual activities will provide a basis for evaluation,

Participation	10%
Homework	20%
Project	10%
Quizzes	10%
Exams	30%
Best of average homework, quizzes, and exams	10%
Lab writeups	10%

Some work may be graded by someone other than the instructor. However, any questions or concerns about grading should only be directed to the instructor.

6.1 Extra credit

Successfully undertaking an activity for extra credit will add 1/2 of a percentage point to your final score. I will propose opportunities for extra credit throughout the semester, and you may propose ideas as well. I may limit this form of extra credit to 4 points.

To earn this extra credit, you must send me an email with a paragraph summarizing what you learned *within one (1) week of the event*. Do not send a summary of the content. It is more interesting for me to read your reflections on the content than a recapitulation of it.

6.2 Good-faith grade guarantee™

Because I realize that computer science does not "click" will all students, I reward effort as well as outcome. Hence, students who make a good faith effort in this class will pass the class with at least a *C*. A good-faith effort includes:

- missing no more than two classes,
- turning in every homework assignment, and
- spending the requisite time on each examination.

7 Attendance

Because is a collaborative, discussion-based course, your presence is integral to your learning. Thus, 1.5% will be deducted from your *overall* grade for each absence. I know that sometimes "things happen." Therefore, you will be granted **one** unexcused absence from class without penalty. However, this one-time rebate is void upon a second absence.

If you are absent, I would appreciate a written explanation (email is appropriate). If you know in advance that you will be absent for any reason, please notify me in writing (again, email is fine) at least 7 days in advance to make arrangements for considering your absence excused.

Because I do not wish you to risk harm to yourself or others, I am likely to moderate penalties in case of illness.

If you do miss a class, you must first talk to a classmate about any material that you may have missed. After that, you may follow up with me about any further questions or concerns.

You should complete the lab assigned for any days you are absent and be sure you understand the material.

8 Academic honesty

As students, you are members of the academic community. Both the College and I expect the highest standards of academic honesty. (See the Grinnell College Student Handbook, e.g., http://www.grinnell.edu/offices/studentaffairs/shb/academicpolicies/academichonesty). Among other things, this means clearly distinguishing between work that is your own, and work that should be attributed to others. This includes ideas and examples that you draw from labs and readings.

It is expected that the collaboration policies given in this syllabus and on particular assignments will be followed. In particular:

- When you explicitly work as part of a group or team, you need not identify the work of each individual (unless I specify otherwise).
- You may discuss *concepts* (algorithms, ideas, approaches, etc.) described in the readings, lab exercises, or during class with *anyone.*

- You may only discuss homework assignments (algorithms, solutions, write-ups, code, debugging, etc.) with your group members, computer science tutors, CSC 151 mentors, or the CSC 151 instructors.
- All the work submitted (code, experimental data, write-ups, etc.) must be your own or that of your group. Code or documentation provided by the instructor must be attributed. No other code or written work (from *any* source) may be shared with others or copied for your own use.
- All non-syntax consultations (i.e., ideas about algorithms) from any source, including the readings, labs, provided code, and internal or external language references, require formal citation within the related program or write-up.
- **Any conceptual contributions by individuals not in your group must be acknowledged and attributed in your report.** That is you must give specific attribution for **any** assistance you receive. (This includes from tutors or mentors.) The suggested acknowledgment format is

  "*[Person X] helped me to do [thing Y] by [explaining Z].*"

- Any program results or output must be faithfully recorded, not forged. (A thoughtful explanation of unexpected behavior can often be a worthwhile submission and is *much* better than the alternative.)
- You are responsible for safeguarding your work from being copied by others. This requires you to take reasonable precautions with hard copy printouts as well as file system permissions. (Note that MathLAN's default permissions prevent others from viewing your files.)

As an instructor, I will meet my obligation to bring any work suspected to be in violation of the College's Academic Honesty Policy to the attention of the Committee on Academic Standing, after which there is no recourse with me.

9 Deadlines

Work is due at the time and date specified in the assignment. Each calendar day your work is late will reduce your grade by one level (exams excluded). Work must still be submitted by the due date if you have arranged a prior excused absence.

Because I am concerned about your health and well being, I may also accept late work (exams excluded) if

1. you start the assignment **at least three days** in advance of the due date;
2. you expend a reasonable amount of effort to complete the assignment by midnight;
3. you send me an e-mail attesting to facts 1 and 2 with whatever work you'd completed when the assignment is due;
4. you go to sleep after sending that e-mail; and
5. you make an appointment to talk with me immediately about any problems you've had on the assignment.

Deadlines for assignments involving programming will automatically be extended by at least one class period if MathLAN is down for an unscheduled period of 3 or more hours during the two days preceding the assignment due date.

Absolute deadline: All work (except the final) must be submitted by Monday 12 December (Note that this is earlier than the institutional deadline.)

10 Getting help

The Math Lab makes tutors for 151 available for help in the open laboratory, SCI 3815. These tutors can be found at regularly scheduled times, which are posted on the front door. Our student tutors may also be available for regular, more intensive one-on-one tutoring. As the course gets underway, please let me know if you are interested.

Of course, you can also get help from me. Please come by during my office hours to discuss the course content, get any extra assistance, or just talk about how the course is going. Note that if multiple students have similar questions or issues, we may work together as a group.

If you cannot attend a scheduled office hour, you may also email me to schedule an appointment; please include 3-4 possible meeting times so that I can find one that works for both of us.

I enjoy getting to know my students, but I prefer to reserve office hours for academic matters. If you would like to have a more informal conversation, I would be delighted to accept an invitation to eat lunch with you at the Marketplace.

Email is also a reliable way to contact me, but please allow 24 hours for a response (except on weekends, when I do not regularly read email). You may also call me in my office (x9812) for more urgent matters (e.g., you will be missing a lab due to illness).

Thanks to Janet Davis, Sam Rebelsky, and Henry Walker for many elements of this syllabus.