

CSC151.02 2014S, Class 17: Exam 1 Discussion

Overview

- Preliminaries.
 - Admin.
 - Questions.
- General issues.
- Exam strategies.
- Additional Q&A.

Preliminaries

Admin

- A remote hello to my grad school colleague Frank Corley. Will he find this message when he next Googles his name?
 - Frank: Thanks for sending good kids to Grinnell.
- Reading for Tuesday: Boolean Values and Predicate Procedures
- Exam 1 returned. It will be the topic of today's class.
 - Yes, I've been known to screw up on grading. Let me know if you have questions.
 - Yes, my handwriting is awful, particularly after I've graded the first dozen exams. Let me know when you can't read it.
 - I've distributed some notes on exam 1 via email.
Please read them before you talk to me about the exam.
 - Please visit with me if you invoked "there's more to life".
 - I do not report statistics on exams.
- I expect that there are some questions about the homework, and have reserved time for those questions.
- Extra credit:
 - Study Break tonight at 8pm
 - Improv show Saturday
 - Thursday extra: Technical interviews

Questions on the Homework

How do I get the `image-transform` problem to work?

`image-variant` needs a one-parameter procedure. You have a two-parameter procedure. You will probably need an anonymous procedure to fill in the second parameter.

```
(define myproc
  (lambda (color some-other-parameter-probably-named-gamma)
    ...))

(image-variant image (lambda (color) (myproc color some-other-value)))
```

What is an anonymous procedure?

A procedure with no name. It typically has the form `(lambda (val) ...)`.

How do we take lists as input?

The same way you take any value as input.

```
(define myfun
  (lambda (lst)
    ...))
```

Suppose I have a procedure that I've taken as input and I want to apply it to a value, how do I do so?

```
(proc value)
```

Sam, that wasn't very helpful. Can you be a bit more detailed.

Well ... part of the goal of the problem is that you figure this out. But an anonymous procedure that takes your procedure as input would help

```
(lambda (proc) (proc ...))
```

Apply some procedure to 1, 2, and 3 and make a list of the result

```
(map (lambda (proc) (list (proc 1) (proc 2) (proc 3)))
     (list square increment))
```

General issues

- No exam statistics!
- A bad approach to the exam? Don't banga
- It's a lot easier to know if a procedure seems to be correct than if documentation or a test seem to be sufficient.
- Limited vision is insufficient

Exam strategies

Our approach:

- Individually, make a list of 3 "Do"s or "Don't"s.
- Share with a group of four (or so).
- Share with class.

Do:

- Regularly send yourself a copy of the exam so that when DrRacket crashes you haven't lost everything.
 - Maybe reate a separate .rkt file for every problem.
 - Long-term: Learn how to use git to preserve all versions of your files.
- Review your work *during work time*, so that you understand it when you hit the exam.
- Constantly run experiments to make sure that your code works.
 - Or write unit tests in advance.
- Read through your work a few times, particularly documentation.
- Compare your work to the provided examples.
- Start early, maybe as soon as Sam posts the test.
 - Lets you see what the test includes.
 - Might let you pace yourself.
 - Your subconscious is sometimes a very good problem solver.
 - And we sometimes have related problems on labs and quizzes.
- Work on a problem for a little while and then move on to the next one if you haven't solved it.
- Take a lot of breaks.
- Pay attention to subtle phrasing.
- Challenge Sam when his instructions are unclear.
- Read the error messages. They do give you some hint as to what's wrong.
- You do better on your work if you have proper nutrition.
- Use the various paper resources
 - Your homework
 - Your labs
 - The readings and labs
 - Maybe the eboards

Don't:

- Spend too much time on one problem.
- Try to solve every problem "all at once".
- Try to solve the test in one sitting.
- Miss anything in the instructions.
- Turn your test in too early.
- Start too early. Let the gunners find the bugs in the exam.

Additional Q&A

Samuel A. Rebelsky, rebelsky@grinnell.edu

Copyright (c) 2007-2014 Janet Davis, Samuel A. Rebelsky, and Jerod Weinman. (Selected materials are copyright by John David Stone or Henry Walker and are used with permission.)



This work is licensed under a Creative Commons Attribution 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/3.0/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.