

Final Examination

Distributed: Friday, May 11, 2007

Due: noon, Friday, May 18, 2007

No extensions.

This page may be found online at

<http://www.cs.grinnell.edu/~rebelsky/Courses/CS302/2007S//Exams/final.html>.

This exam is also available in PDF format.

Contents

- Preliminaries
- Problems
 - Problem 1: Revising CSC302
 - Problem 2: Varying Approaches to Introspection
 - Problem 3: Polymorphism
 - Problem 4: Presentations
 - Problem 4.a: Garbage Collection
 - Problem 4.b: Prolog
 - Problem 4.c: PostScript
 - Problem 4.d: Scripting Languages
- Citations
- Some Questions and Answers
- Errors

Preliminaries

There are four problems on the exam. Some problems have subproblems. Each problem is worth the same number of points. The point value associated with a problem does not necessarily correspond to the complexity of the problem or the time required to solve the problem.

Experience shows that different people find different problems complex. Hence, if you get stuck on an early problem, try moving on to another problem and let your subconscious work on the early problem. (You'll also get a better sense of accomplishment if you can find at least one problem that you can solve early.)

This examination is open book, open notes, open mind, open computer, open Web. However, it is closed person. That means you should not talk to other people about the exam. Other than as restricted by that limitation, you should feel free to use all reasonable resources available to you. As always, you are expected to turn in your own work. If you find ideas in a book or on the Web, be sure to cite them appropriately.

Although you may use the Web for this exam, you may not post your answers to this examination on the Web (at least not until after I return exams to you). And, in case it's not clear, you may not ask others (in person, via email, via IM, by posting a "please help" message, or in any other way) to put answers on the Web.

This is a take-home examination. You may use any time or times you deem appropriate to complete the exam, provided you return it to me by the due date.

I expect that someone who has mastered the material and works at a moderate rate should have little trouble completing the exam in a reasonable amount of time. In particular, this exam is likely to take you about eight hours, depending on how well you've learned topics and how fast you work. *You should not work more than twelve hours on this exam. Stop at twelve hours and write "There's more to life than CS" and you will earn at least 70 points on this exam.*

I would also appreciate it if you would write down the amount of time each problem takes. Each person who does so will earn two points of extra credit. Since I worry about the amount of time my exams take, I will give two points of extra credit to the first two people who honestly report that they've spent at least six hours on the exam or completed the exam. (At that point, I may then change the exam.)

You must include both of the following statements on the cover sheet of the examination. Please sign and date each statement. Note that the statements must be true; if you are unable to sign either statement, please talk to me at your earliest convenience. You need not reveal the particulars of the dishonesty, simply that it happened. Note also that "inappropriate assistance" is assistance from (or to) anyone other than Professor Rebelsky (that's me) or Professor Davis.

1. I have neither received nor given inappropriate assistance on this examination.
2. I am not aware of any other students who have given or received inappropriate assistance on this examination.

Because different students may be taking the exam at different times, you are not permitted to discuss the exam with anyone until after I have returned it. If you must say something about the exam, you are allowed to say "This is among the hardest exams I have ever taken. If you don't start it early, you will have no chance of finishing the exam." You may also summarize these policies. You may not tell other students which problems you've finished. You may not tell other students how long you've spent on the exam.

You must present your exam to me in two forms: both physically and electronically. That is, you must write all of your answers using the computer, print them out, number the pages, put your name on the top of every page, and hand me the printed copy. You must also email me a copy of your exam. You should create the emailed version by copying the various parts of your exam and pasting them into an email message. In both cases, you should put your answers in the same order as the problems. Failure to name and number the printed pages will lead to a penalty of two points. Failure to turn in both versions may lead to a much worse penalty.

In many problems, I ask you to write code. Unless I specify otherwise in a problem, you should write working code and include examples that show that you've tested the code.

Just as you should be careful and precise when you write code and documentation, so should you be careful and precise when you write prose. Please check your spelling and grammar. Since I should be equally careful, the whole class will receive one point of extra credit for each error in spelling or grammar you identify on this exam. I will limit that form of extra credit to five points.

I will give partial credit for partially correct answers. You ensure the best possible grade for yourself by emphasizing your answer and including a *clear* set of work that you used to derive the answer.

I may not be available at the time you take the exam. If you feel that a question is badly worded or impossible to answer, note the problem you have observed and attempt to reword the question in such a way that it is answerable. If it's a reasonable hour (before 10 p.m. and after 8 a.m.), feel free to try to call me in the office (269-4410) or at home (236-7445).

I will also reserve time at the start of classes next week to discuss any general questions you have on the exam.

Problems

Problem 1: Revising CSC302

Topics: “The Big Picture”, Primary literature

As you know, one important focus of this course is to help you develop the skills to read “the literature”, that is, the papers in journals and conferences that provide the central ideas of computer science in general and programming languages in specific. I help you develop those skills by having you read and respond to a wide variety of papers.

Each semester, I update my list of papers to use in the course. For this question, you have the opportunity to affect that choice.

- a. Find a paper from the *research* literature that would be appropriate to include in CSC302. When I say “research literature”, I mean that you should focus on articles from journals and conferences, rather than on Web pages or on manuals and such. Obtain a copy of that paper for yourself and make a copy for me. (You should submit my copy along with the exam.) If you need help getting a paper whose title you have found, feel free to ask me for help.
- b. In a paragraph or two, describe the state of the field of programming languages at the time the paper was written.
- c. Summarize the thesis of the paper.
- d. In two or three paragraphs, explain how including this paper in the CSC302 syllabus would benefit the course.

Problem 2: Varying Approaches to Introspection

Topics: Introspection, Java, Ruby

One technique we explored this semester was *introspection* (closely related to *reflection*), in which we can write programs that can take apart classes and objects to look at the internal issues.

In four or so paragraphs, explain the similarities and differences between the implementation of introspection in Java and the implementation of introspection in Ruby. Make sure to use sample code in your explanations.

Problem 3: Polymorphism

Topics: Types, Polymorphism, Java, Scheme, C

In “On Understanding Types, Data Abstraction, and Polymorphism”, Cardelli and Wegner identify four kinds of polymorphism, two universal (parametric and inclusion) and two ad hoc (overloading and coercion).

- a. Which of these four kinds of polymorphism does Scheme support? For each kind that Scheme supports, provide an illustrative example.
- b. Which of these four kinds of polymorphism does Java support? For each kind that Java supports, provide an illustrative example.
- c. Which of these four kinds of polymorphism does C support? For each kind that C supports, provide an illustrative example.

Problem 4: Presentations

Topics: Garbage Collection, Prolog, PostScript, Scripting Languages

Problem 4.a: Garbage Collection

An accompanying diagram provides a simple illustration of a state of memory during the execution of a program. In this diagram, each grey square represents one cell of memory. The small number in the lower-left-hand-corner of each cell is that cell’s memory location. The larger number in each cell is the contents of that cell. For the purposes of this problem, you should assume that each cell contains a pointer. The darker squares and rectangles represent objects.

For example, a length-two object resides in cells 26 and 27, and contains pointers to the length-three object in cells 10-12 and the length-one object in cell 30.

In this problem, you will explore the effect of garbage collecting this memory using each of three algorithms. For each collection, you can assume that the root set contains only the address 25. (You may also use the address 15, which appeared in an earlier version of the exam. Amazingly, the two addresses give the same live set.)

i. Draw the state of memory after a mark-and-sweep collector cleans up. You may find it useful to use this diagram of empty memory. Although mark-and-sweep collectors do not clear cells, you should leave any cells that are available blank.

ii. Draw the state of memory after a mark-and-compact collector cleans up. You may find it useful to use this diagram of empty memory. Although mark-and-compact collectors do not clear cells, you should leave any cells that are available blank.

iii. Draw the state of memory after a simple semispace collector cleans up. You may find it useful to use this diagram of a separate space in memory. Although semispace collectors do not clear cells, you should leave any cells that are available blank.

Problem 4.b: Prolog

The following `remove_first` predicate is used to determine whether a list can be created by removing one value from another list. In particular, `remove_first(X,Lst,NewLst)` holds if `NewLst` is the same as `Lst`, but with the first instance of `X` removed.

```
remove_first(X,[],[]).
remove_first(X,[X|Xs],Ys) :- equal_lists(Xs,Ys).
remove_first(X,[Y|Ys],[Y|Zs]) :- neq(X,Y),remove_first(X,Ys,Zs).
equal_lists([],[]).
equal_lists([X|Xs],[X|Ys]) :- equal_lists(Xs,Ys).
neq(a,b). neq(a,c). neq(a,d).
neq(b,a). neq(b,c). neq(b,d).
neq(c,a). neq(c,b). neq(c,d).
neq(d,a). neq(d,b). neq(d,c).
```

For example,

```
| ?- remove_first(a,[a,b,c],[b,c]).
true ?
yes
| ?- remove_first(b,[a,b,c],[a,c]).
true ?
yes
| ?- remove_first(c,[a,b,c],[a,b]).
true ?
yes
| ?- remove_first(d,[a,b,c],[a,b,c]).
true ?
yes
| ?- remove_first(a,[a,b,c],[a,b,c]).
no
| ?- remove_first(a,[a,b,a,c],[b,a,c]).
true ?
yes
| ?- remove_first(a,[a,b,a,c],[b,a,c]).
true ?
yes
| ?- remove_first(a,[a,b,a,c],[a,b,c]).
no
```

- i. Explain why we use the `neq` predicate in the third portion of the definition of `remove_first`.
- ii. Explain what happens when you provide a variable for the third parameter of `remove_first`. For example, why do you get the results you get when you ask for `remove_first(a, [a,b,a,c], X)`?
- iii. Explain what happens when you provide a variable for the second parameter of `remove_first`.
- iv. Explain what happens when you provide variables for the first and third parameters of `remove_first`.

Problem 4.c: PostScript

I've written a PostScript procedure, `linebetween`, that draws a line between the two points on the top of the stack. For example, if we have `10 20 100 200` on the stack and then call `linebetween`, PostScript will draw a line between the points (10,20) and (100,200). The `linebetween` procedure also removes the four values from the stack.

Here's a simple PostScript program to draw a house using `linebetween`. You can see it in action by saving it in a file with a `.ps` suffix and then opening the file with `ghostview`, which is available through the `gv` command.

```

%!PS-Adobe-3.0
%%BoundingBox: 0 0 612 792
%%LanguageLevel: 1
%%EndComments

0.0 0.0 0.0 setrgbcolor
10 setlinewidth

/linebetween { moveto lineto stroke } def

100 100 200 100 linebetween
100 100 100 180 linebetween
200 100 200 180 linebetween
 75 155 150 230 linebetween
225 155 150 230 linebetween

showpage

%%EOF

```

Write and test a PostScript procedure, `rectangle` that draws a rectangle, assuming the stack contains the x coordinate of the left, the y coordinate of the bottom, the width, and the height of the rectangle. For example, the following should draw a rectangle with corners of (100,200), (150,200), (150,300), and (100,300).

```
100 200 50 100 rectangle
```

In writing rectangle, you may use only linebetween and the following PostScript stack manipulation procedures: add, dup, exch, index, pop, and roll. Your procedure should not leave any values on the stack.

Problem 4.d. Scripting Languages

i. In two or three paragraphs, argue that Scheme is a scripting language, using Ousterhout's discussion of scripting languages as your primary basis.

ii. In two or three paragraphs, argue that Scheme is not a scripting language, using Ousterhout's discussion of scripting languages as your primary basis.

Citations

Cardelli, Luca & Wegner, Peter. (1985). On Understanding Types, Data Abstraction, and Polymorphism. *ACM Computing Surveys* **17**(4), December 1985.

Ousterhout, John K. (March 1988). Scripting: Higher-level programming for the 21st century. *IEEE Computer* **31**(3), pp. 23-30.

Some Questions and Answers

These are some of the questions students have asked about the exam and my answers to those questions.

Problem 1

Problem 2

Problem 3

Problem 4

Errors

Here you will find errors of spelling, grammar, and design that students have noted. Remember, each error found corresponds to a point of extra credit for everyone. I usually limit such extra credit to five points. However, if I make an astoundingly large number of errors, then I will provide more extra credit.

- Spurious "at the implementation" in problem 2. [EO'N, 1 point]
- In problem 2, "you explantations" should be "your explanations". [EB, 1 point]
- "Paragraph of two" should be "Paragraph *or* two". [MU/DD, 1 point]
- "the the" appears at least once (or did, before the error was caught). [JW/EO'N, 1 point]
- PDF link goes to midsem. [JT, 1 point]
- ghostview is gv in the MathLAN. [JT, 1 point]
- When changing the addresses in the illustration, Sam forgot to change the root address in the code. That root address should be 25, rather than 15. [EO, 1 point]