

Homework 6: Writing Procedures

Assigned:

Due: (changed due to MathLAN failure)

No extensions!

Summary: In this assignment, you will practice writing your own procedures for a variety of tasks.

Purposes: To give you experience writing procedures. To provide you with some feedback on the expected “style” of writing procedures.

Expected Time: One hour.

Collaboration: You may work in a group of any size between one and four, inclusive. You may consult others outside your group, provided you cite those others. You need only submit one assignment per group.

Submitting: Email me your work, using a subject of *CSC151 Homework 6*.

Warning: So that this exercise is a learning assignment for everyone, I may spend class time publicly critiquing your work.

Assignment

The focus of this assignment is a series of small problems, rather than a larger problem. Write procedures to solve each of the problems below. Since this is an exercise in writing procedures, and not in documenting procedures, you are not required to write documentation (although you may earn extra credit for writing good documentation).

However, one of the purposes of this assignment is to encourage good style, so you should think about how to structure and format your procedure definitions to make them easy to read and understand.

Problem 1: Swapping List elements

Write a procedure, (`swap-first-two lst`), that, given a list as an argument, creates a new list that interchanges the first two elements of the original list, leaving the rest of the list unchanged. Thus,

```
> (swap-first-two (list 'a 'b 'c 'd 'e))  
(b a c d e)
```

In this problem, assume that the list given to `swap-first-two` has at least two elements; do not worry about the possibility that `swap-first-two` might be applied to numbers, symbols, empty lists, or lists with only one element.

Problem 2: Funding the Rosenfield Center

As you may have noted, the coffee prices at the new Grill (no “e”) are clearly intended to help the College recoup cost overruns on the Joseph Rosenfield ’25 Center. (Disclaimer: The preceding sentence was intended as a joke and not as a critique of the College, the administration of the College, of the JR25C, or of Dining Services.)

Suppose that the Grill charges \$1.75 for a small cup of coffee and \$2.90 for a large cup of coffee.

Write a procedure, (`coffee-proceeds small large`) that, given the numbers of small and large coffees purchased, returns the gross income of the Grill from coffee *in cents*. For example,

```
> (coffee-proceeds 10 0)
1750
> (coffee-proceeds 0 8)
2320
> (coffee-proceeds 83 51)
29315
```

Problem 3: From Proceeds to Profits

Suppose the it costs five cents to make an ounce of coffee, that a small cup of coffee holds twelve ounces, and that a large cup of coffee holds twenty ounces. Write a procedure, (`coffee-profits small large`), that computes the profit that the Grille makes upon selling the specified numbers of small and large cups of coffee. The result should be in cents. For example,

```
> (coffee-profits 10 0)
1150
> (coffee-profits 0 8)
1520
> (coffee-profits 83 51)
19235
```

Problem 4: Indefinite Articles

One problem that programmers often encounter in trying to write programs like the Mad Libs[®] game of homework 5 is getting appropriate articles to precede nouns and noun phrases. For example, if the template is (`string-append name " ate an " fruit "."`), then when someone uses a fruit that does not begin with a vowel (e.g., "banana"), the sentence has in correct grammar. If the template uses “a” rather than “an”, the program has problems if the fruit does begin with a vowel. Some make the compromise of writing “a(n)”, but that’s just ugly.

What’s the solution? A procedure that determines the correct indefinite article to prefix a string.

Write a procedure, (`indefinite-article noun-phrase`), that returns a string representing the appropriate indefinite article with which to precede *noun-phrase*.

For example,

```
> (indefinite-article "banana")
"a"
> (indefinite-article "apple")
"an"
> (indefinite-article "indescribably good banana")
"an"
> (indefinite-article "big apple")
"a"
> (string-append (indefinite-article "big apple") " " "big apple")
"a big apple"
> (string-append (indefinite-article "orange banana") " " "orange banana")
"an orange banana"
```

It is up to you whether you use American or English rules for words that begin with an “h”.

```
> (string-append (american-indefinite-article "historical event") " " "historical event")
"a historical event"
> (string-append (british-indefinite-article "historical event") " " "historical event")
"an historical event"
```

Problem 5: Limiting Precision

For some numbers, the `exact->inexact` procedure produces lots and lots of digits after the decimal point.

```
> (exact->inexact (/ 1 3))
0.3333333333333333
> (exact->inexact (/ 22 7))
3.142857142857143
```

In many cases, it is preferable to have just two digits after the decimal point. Write a procedure, `(approximate num)`, that represents `num` with no more than two digits after the decimal point.

```
> (approximate 1/3)
0.33
> (approximate 22/7)
3.14
> (approximate 5)
5.0
> (approximate 1.249)
1.25
> (approximate 1.246)
1.25
> (approximate 1.254)
1.25
```

You may need to use multiplication, division, `round`, `exact->inexact` and other numeric procedures.

Important Evaluation Criteria

Students who provide correct expressions for each question will earn a check.

Students who provide oddly formatted or inelegant solutions to the problems will be publicly critiqued for their odd formatting and inelegance, but will not receive a grade penalty.

Students who provide particularly elegant formatting or strategies will earn a higher grade.

Copyright © 2007 Janet Davis, Matthew Kluber, and Samuel A. Rebelsky. This material is based upon work partially supported by the National Science Foundation under Grant No. CCLI-0633090. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This work is licensed under a Creative Commons Attribution-NonCommercial 2.5 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/2.5/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.