Fundamentals of Computer Science I: Media Computing (CS151.02 2007F)

Laboratory: The DrFu Programming Environment

Summary: In this laboratory, you will begin to type Scheme expressions in DrFu. Scheme is the language in which we will express many of our algorithms this semester, and DrFu is the environment in which we will write those expressions.

Contents:

Introduction:

Many of the fundamental ideas of computer science are best learned by reading, writing, and executing small computer programs that illustrate them. One of our most important tools for this course, therefore, is a *program-development environment*, a computer program designed specifically to make it easier to read, write, and execute other computer programs. In this class, we will often use a programming environment named DrFu. As you may recall from the introductory Linux laboratory, DrFu is a locally-developed environment that applies the interface of DrScheme to programming for The GIMP.

Preparation

If you successfully completed the Linux laboratory, you should have an icon for DrFu at the bottom of your screen. (The icon is currently a hastily scrawled red lambda with a pen near the top, but that may change.) Click on that icon to start DrFu. A few GIMP windows should appear. A DrScheme opening screen should also appear. After you wait a few minutes, you should see a DrScheme editing window appear. You will do your primary work in this window.

Exercises

Exercise 1: Discovering DrScheme's Interactions Pane

Short Version

- The lower text area is called the *interactions pane*.
- You type scheme commands there and DrScheme responds.
- Type the first few examples from the reading.
  - `(sqrt 144)`
  - `(+ 3 4)`
  - `(+ 3 (* 4 5))`
  - `(* (+ 3 4) 5)`
  - `(string-append "Hello" " " "Sam")`

As you may remember from the reading on DrFu, the DrScheme window has two panes, one for definitions and one for interaction. Just as in the reading, we'll begin by considering the interactions panel.

The best way to understand the interactions pane is to use it. So, let's try the first few examples from the reading. Type each in the pane, hit return, and see if you get the same value.

```
> (sqrt 144)
12
> (+ 3 4)
7
> (+ 3 (* 4 5))
23
> (* (+ 3 4) 5)
35
> (string-append "Hello" "Sam")
HelloSam
```

Exercise 2: Reflection: How Do You Know It's Correct?

Short Version

- Try typing `(sqrt 137641)`.
- Reflect on how you know whether or not the answer is correct.

Of course, the computer is using some algorithm to compute values for the expressions you enter. How do you know that the algorithm is correct? One reason that you might expect it to be correct is that DrScheme and the GIMP are is widely-used programs (and ones that I've asked you to use). However, there are bugs even in widely-used programs. You may recall a controversy a few years back in which it was discovered that a common computer chip computed a few specific values incorrectly, and no one had noticed. In addition, you know that some Grinnell students have hacked together DrFu, our bridge between DrScheme and the GIMP, so you might be a bit suspicious.

Each time you do a computation, particularly a computation for which you have designed the algorithm, you should consider how you might verify the result. (You need not verify every result, but you should have an idea of how you might do so.) When writing an algorithm, you can then also use the verification process to see if your algorithm is right.

Let's start with a relatively simple example. Suppose we ask you to ask DrScheme to compute the square root of 137641. You should be able to do so by entering an appropriate Scheme expression:

```
> (sqrt 137641)
```

Scheme will give you an answer. How can you test the correctness of this answer? What if you don't trust DrScheme's multiplication procedure? (Be prepared to answer this question for the class as a whole.)

Exercise 3: DrScheme's Definitions Pane

Short Version

- The interactions pane is temporary. The top pane, called the *definitions pane* is more permanent.
- When you click Run, the interactions pane is erased and the instructions in the definitions pane are executed.
- Enter the definitions from the reading.
- Ask DrFu to compute the average and maximum grade.

As you may recall from the reading, the upper text area in the DrScheme window, which is called the *definitions pane*, is used when you want to prepare a program "off-line", that is, without immediately executing each step. Instead of processing what you type line by line, DrScheme waits for you to click on the button labelled Run (the second button from the right, in the row just below the menu bar) before starting to execute the program in the definitions pane. If you never click on that button, fine -- your program is never executed.

Warning: When you click on the Run button, the contents of the interactions pane are erased. The idea is that executing the program in the definitions pane may invalidate the results of previous interactions. Erasing the results that may now be inconsistent with the new definitions ensures that all visible interactions use the same vocabulary. This is actually a helpful feature of DrScheme, but it can take you by surprise the first time you see it happen. Just make sure that you have everything you need from the interactions pane before clicking on Run.

Let's try using the definitions pane. First, enter the following in that pane.

```
(define grade1 90)
(define grade2 67)
(define grade3 80)
(define grade4 85)
(define grade5 100)
(define grade6 91)
```

Next, try computing the average in the interactions pane.

```
> (/ (+ grade1 grade2 grade3 grade4 grade5 grade6) 6)
reference to undefined identifier: grade
```

It is likely that you will get an error message, just as the example suggests. Why? (Please make sure you have an answer before going on.)

Next, click Run and try entering the expression again. (Remember, <Esc>+<P> will bring back the previous expression.)

Exercise 4: Definitions, Revisited

Let's try another definition. Define `name` as your name in quotation marks. For example,

```
(define name "SamR")
```

Click <Run> and then find the value of the following expression.

```
> (string-append "Hello " name)
```

Exercise 5: Saving to Files

Let's make sure that you can save and restore the work you do in the definitions pane.

- Save your definitions as `/home/`*`username`*`/Desktop/grades.sct`.
- Quit DrScheme and The Gimp.
- Restart DrFu.
- Open `/home/`*`username`*`/Desktop/grades.sct`.
- Click <Run>
- In the interactions pane, enter the expressions from above
  - `(/ (+ grade1 grade2 grade3 grade4 grade5 grade6) 6)`
  - `(string-append "Hello " name)`

Exercise 6: Loading Definitions

Let's try using the definitions you created without having them open in the definitions pane.

- Add a new grade, `grade7`, to `grades.sct` and save the file, but do not run it.
- Open a new DrScheme window by selecting New from the File menu.
- In the interactions pane of the new window, type `grade7`. You should get an error message, which tells you that `grade7` is not yet defined.
- In the interactions pane of the new window, type
  - `(load "/home/`*`username`*`/Desktop/grades.sct")`
- In the interactions pane, type `grade7`. You should now see a value.

In the future, we will be creating some .sct files that we change infrequently. Those files we will typically load, rather than open.

Exercise 7: A Library

Throughout the semester, you will find yourself defining a number of values (including algorithms, which Scheme considers values). So that you don't have to go back to lots of places to look for things you've done before, I suggest that you create a file, `library.sct` in which you enter values and other stuff that you'll need again and again.

Create that file, and give it two values, `first-name`, which should have the value of your first name, and `last-name`, which, should have the value of your last name.

For example,

```
(define first-name "Sam")
(define last-name "Rebelsky")
```

Exercise 8. Other Notations

As you've learned, Scheme expects you to use parentheses and prefix notation when writing expressions. What happens if you use more traditional mathematical notation? Let's explore that question.

Type each of the following expressions at the Scheme prompt and see what reaction you get.

1. `(2 + 3)`
2. `7 * 9`
3. `sqrt(49)`

You may wish to read the notes on this problem for an explanation of the results that you get.

Exercise 9: Showing Images

So far, it's been hard to tell that DrFu is associated at all with the GIMP. However, you can use DrFu to control the GIMP in a variety of ways. We'll start with a simple one. We will load an image and then show that image. The procedure `image.load` loads an image and returns an integer that DrFu can use to reference the image. The procedure `image.show` shows that image.

a. Tell DrFu to load the image `"/home/rebelsky/glimmer/samples/rebelsky-stalkernet.jpg"`. When you do so, you should see a number. Make a note of that number.

b. Tell DrFu to show that image.

c. Save your StalkerNet picture (or a friend's StalkerNet picture) to your desktop.

d. Tell DrFu to load that image. The full name of the image will be something like "/home/*yourusername*/Desktop/*nameoffile*.jpg".

e. Tell DrFu to show that image.

f. As you may have observed, it's a pain to have to type in the full path of a file. Typically, we write definitions to avoid retyping.

Use `define` to give the name `sams-picture` to my StalkerNet photo and the name `my-photo` to your StalkerNet photo.

For Those with Extra Time

Extra 1: Grading

Let's play for a bit with how one might use DrScheme to compute grades. (We teach you this, in part, so that you can figure out your estimated grade in this class and others.) Let's define five names, `grade1` through `grade5` that potentially represent grades on five homework assignments.

```
(define grade1 95)
(define grade2 93)
(define grade3 105)
(define grade4 30)
(define grade5 80)
```

Looking at those grades, you might observe that the student seems to have spent a bit of extra work on the third assignment, but that the extra work so disrupted the student's life that the next assignment was a disaster. (You may certainly analyze the grades differently.)

a. Write a definition that assigns the name `average-grade` to the average of the grades without dropping the highest and lowest grades.

Many faculty members discard these "outliers", with a grading policy of "I take the average of your grades after dropping the highest grade and the lowest grade".

b. Write a definition that assigns the name `highest-grade` to a *computed* highest grade. (That is, `highest-grade` should remain correct, even if I change the values associated with `grade1` through `grade5`.) In writing this definition, you may find the `max` procedure useful.

c. Write a definition that assigns the name `lowest-grade` to a *computed* lowest grade. You may find the `min` procedure useful.

d. Write a definition that assigns the name `weighted-average` to the weighed average grade (that is, the grade that results from dropping the lowest and highest grades and then averaging the result).

Extra 2: Definitions, Revisited

As you observed in the DrScheme lab, you can use the definitions pane to name values that you expect to use again (or that you simply find it more convenient to refer to with a mnemonic). So far, all we've named is simple values. However, you can also name the results of expressions.

a. In the definitions pane, write a definition that assigns the name `seconds-per-minute` to the value 60.

b. In the definitions pane, write a definition that assigns the name `minutes-per-hour` to the value 60.

c. In the definitions pane, write a definition that assigns the name `hours-per-day` to the value 24.

d. In the definitions pane, write a definition that assigns the name `seconds-per-day` to the product of those three values. Note that you should use the following expression to express that product.

```
(* seconds-per-minute minutes-per-hour hours-per-day)
```

e. Run your definitions and confirm in the interactions pane that `seconds-per-day` is defined correctly.

f. Add these four definitions to the `library.ss` file you created earlier.

Notes on the Problems

Notes on Exercise 8: Other Notations

```
(2 + 3)
```

When DrScheme sees the left parenthesis at the beginning of the expression `(2 + 3)`, it expects the expression to be a procedure call, and it expects the procedure to be identified right after the left parenthesis. But `2` does not identify a procedure; it stands for a number. (A "procedure application" is the same thing as a procedure call.)

```
7 * 9
```

In the absence of parentheses, DrScheme sees `7 * 9` as three separate and unrelated expressions -- the numeral `7`; `*`, a name for the primitive multiplication procedure; and `9`, another numeral. It interprets each of these as a command to evaluate an expression: "Compute the value of the numeral `7`! Find out what the name `*` stands for! Compute the value of the numeral `9`!" So it performs the first of these commands and displays `7`; then it carries out the second command, reporting that `*` is the name of the primitive procedure `*`; and finally it carries out the third command and displays the result, `9`. This behavior is confusing, but it's strictly logical if you look at it from the computer's point of view (remembering, of course, that the computer has absolutely no common sense).

```
sqrt(49)
```

As in the preceding case, DrScheme sees `sqrt(49)` as two separate commands: `sqrt` means “Find out what `sqrt` is!” and `(49)` means “Call the procedure `49`, with no arguments!” DrScheme responds to the first command by reporting that `sqrt` is the primitive procedure for computing square roots and to the second by pointing out that the number `49` is not a procedure.

Return to the problem.

Working Outside The CS Lab

In the past, it has been possible to run DrScheme at home. It has also been possible to run the GIMP at home. However, the GIMP is currently undergoing revisions, and DrFu relies on some of those revisions. Hence, for the first part of this semester, we ask that students come to the CS wing to do the work for this class. We’ll try to provide a friendly environment for you to work.

Later in the semester, we may be able to provide you with a way to work at home, but that project is still in development.
