CSE 341 - Programming Languages - Autumn 2012

Running Racket

We'll use DrRacket, version 5.3. It is installed on the windows machines in the undergrad labs and on attu, or you can download it for a personal machine.

On the Windows machines in the lab, DrRacket is here:

```
All Programs / DEV TOOLS & LANGUAGES / Racket / DrRacket
```

You can also double-click on a saved Racket source file.

On Linux, start it by typing `drracket` to the shell prompt. You'll need to use XWindows -- DrRacket will come up in a separate window.

Finally, you can download it and install it on a personal machine from http://racket-lang.org/download. It's free.

First Time Set-Up

The first time you use DrRacket, click on "Choose Language" in the lower left and then select "Use the language declared in the source." The text `#lang racket` should appear in the upper definitions pane. Click "Run".

DrRacket should remember this choice henceforth.

Structure of your Racket files

- Create and save programs in the (top) "Definitions Window." You can save your file wherever. Use a .rkt file extension.
- Make the first line of your file exactly (including the # character):

  ```
  #lang racket
  ```

  This tells DrRacket that your file is in the Racket language and not some other language. You can have lines of comments before this line.
- For your homework solution, make the second non-comment line of your file:

  ```
  (provide (all-defined-out))
  ```

  This line is, at least for the time being, working around Racket's module system. In Racket, each file is its own module and this line is making all top-level definitions externally visible, which is not the default. You do *not* need this line to use your definitions in the REPL (the bottom buffer). You *do* need this line (or another approach) to use your definitions from a second testing file.
- For a second testing file, also include the `#lang racket` line and then include the line

  ```
  (require "octopus.rkt")
  ```

  where `octopus.rkt` is the file with the code you want to test. Put both files in the same directory/folder on your computer. Your testing file does not need `(provide (all-defined-out))`.

Using DrRacket

For the most part DrRacket is an easy-to-use system with lots of documentation. Here are a few specific notes related to how we will use it:

- Click "Run" to have the REPL process the definitions in your file.
- Clicking "Run" will "start the REPL over" -- all previous work will disappear and only the definitions from the file you "Run" will be defined.
- If you want to save work in the REPL, do so before clicking "Run." There is also an option under "File" for "Log Definitions and Interactions" but we have little experience with it.
- A useful shortcut in the REPL is to use Alt-P (any number of times) to bring back up recent interactions. Moreover, this works even to bring up interactions from before the most recent "Run." That is, even though the interactions disappeared and are not part of the current environment, you can still use Alt-P to bring them back and then edit them or click Return to re-run them.
- When using a second testing file as described in the previous section, click "Run" in the window that has the testing file. This will send to the REPL your homework solution *and* your tests.
- If your program is stuck in an infinite loop, click "Stop."
- You can (re-)indent a line with the TAB key. You can select multiple lines and hit TAB to reindentall selected lines.
- If you type), DrRacket will replace it with] if doing so will match a [.

Useful Tools

Use the "Debug" button to debug programs, and also to understand how recursive functions are operating. Another useful command is "Check Syntax", which of course checks the syntax, but also lets you see the bindings of variables. (Try hovering the mouse over a variable -- good to help understand lexical scoping.)