COSC 251
Lisp Cheat Sheet
Last Revision: March 22nd, 2013

1 Installation

For Mac & Windows: Download and extract Lispbox from common-lisp.net/project/lispbox (linked on the course page). If you're using Mac, double-click the emacs icon, if you're using Windows, double-click the lispbox.bat icon.

2 Using emacs

Most of the emacs commands are readily available to you from the dropdown menus, but you can navigate using keyboard commands (as it was meant to be used).

Some common commands: (C represents ctrl, M represents the meta key which is alt on Windows, command on Mac)

Command	Result
M-x slime	Starts SLIME if you've closed it
C-x C-f	open or create new file
C-x C-s	save the current file
C-x k	close current buffer
C-x 2	split the buffer into 2 windows
C-x o	change buffer focus once split
C-x 1	remove split
C-x C-right arrow	switch buffers (could use left arrow as well)
C-x C-e	evaluate function ending at this whitespace
C-c C-k	evaluate the entire file
C-x C-c	exit

For both eval requests, switch buffers back to the SLIME interpreter to run those functions. Also, be sure your files have the .lisp extension in order to work in this system correctly.

3 car, cdr, cons

Lisp is a list processing language, as such, everything in lisp, even the commands, are list based. Given a list:

```
1  ; this is a comment
2
3  (defvar l '(1 2 3 4))
4  ; or
5  (defvar l (list 1 2 3 4))
6
7  ;car - first element
8  (car l)
9
10 ;cdr - rest of the list
11 (cdr l)
12
13 ;cons - creation of cons cells
```

```
(cons 1 2)

;cons can be used to append to lists
(cons l 1)
```

4 Variables and operations

```
;let statements, local to the parens for the let only
(let ((x 10) (y 20)) (+ x y))

;defvar statements, global scope
(defvar x 0)

;defparameter statements, global scope, overrides previous binding
(defparameter count 0)

;resetting variable values
(setf x 12)
```

Operations are functions, and as such are formatted in prefix notation (see + in the let example).

5 Control Structures

```
;if statement
;general format:
;(if (cond)
; true clause
; false clause)

(if (<= n 1)
  (+ n 2)
  (+ n 1))

; compound conditionals
(and (>= n 1) (< n 10))
(or (>= n 1) (equal n 5))

;loop - do while
;(do ((variable init update)) (test moreargs) body)
(do ((x 0 (+ x 1))) ((> x 10) 'done) (print x))
```

Note that each clause here is a proper Lisp clause, meaning that inside of a set of parens, you can have as many statements as you need.

6 Functions

```
;basic format of functions
;(defun nameoffunction (parameter list) body)

;basic functions - note return is the last thing executed
```

```
(defun add2ints (n m) (+ n m))

;calling the above function, returns 9
(add2ints 4 5)

;recursive function example - factorial
(defun factorial (n)
  (if (<= n 1)
    1
    (* n (factorial (- n 1)))))

;optional arguments example - will create a list with c = 10 and d = 20
;if those params aren't given
(defun foo (a b &optional (c 10) (d 20)) (list a b c d))

;can key the optional variables to allow for some of them to be user set, and
;others to be left default
(defun foo (a &key (b 10) (c 20) (d 30)) (list a b c d))

;returns (5 10 6 30)
(foo 5 :c 6)

;allowing unlimited parameters
(defun foo (&rest args) (print args))

;prints (1 2 3 4 5 6)
(foo 1 2 3 4 5 6)
```

7 Input and Output

```
; Simple print - more complex printing, use format
(print "Hello, world")

; Simple input - the eval here is for help, not always needed
(setf x (eval (read)))
```