CS151L Fall 2013
Week 9: NetLogo Command Cheat Sheet

Command	Command Description
while **[***conditional reporter***]** **[** *commands* **]**	If the *conditional reporter* (reporter is something that returns a value) reports false, exit the loop. Otherwise run *commands* and repeat. The reporter may have different values for different agents, so some agents may run *commands* a different number of times than other agents. Example: `while [pcolor = black]` ;; while the turtles are on black patches `[ forward 1` `]` ;; move forward 1
let *listname* **[# # # # # ...]** **set** *listname* **[# # # # #...]**	Usually a variable holds only one piece of information. Lists let you store multiple pieces of information in a single variable name (listname) by collecting that information in a list. Each value in the list can be any type of value: a number, or a string, an agent or agentset, or even another list. One way to make a list is by simply putting the values you want in the list between brackets, after the let or set commands Note that the individual values are separated by spaces. If you want to create a local list, the list must be created with the let command and can be changes with the set command. If you want to create a global list or an agent list, then you would define it at the beginning of the program using either the globals command or a turtles-own, <breeds>-own, patches-own or links-own keyword. Example: `let mylist [ 1 5 8 10]` ;; creates a local list called mylist containing [1 5 8 10] `set mylist [2 4 6 8]` ;; changes the value of the local list mylist to [2 4 6 8]
sentence *value1 value2* **...**	Makes a list out of the values. If any value is a list, its items are included in the result directly, rather than being included as a sublist. Example: `set xx [1 2 3 4]` ;; creates a list xx containing [1 2 3 4] `set xx sentence xx [5 6 7]` ;;creates a new list xx containing [1 2 3 4 5 6 7] `show sentence "xx=" xx` ;;very useful in debugging. ;;The show command can only takes one argument. If we want to ;; display two things (in the above example, both the variable name ;; and its current value), then we need to make the two things into one ;; thing. The sentence reporter does exactly that.

`to-report` *procedure-name*
`to-report` *procedure-name* **[***input1* **...]**
	Used to begin a reporter procedure (procedure-name). Can input values into the procedure by using square brackets after the procedure-name. The body of the procedure should use report to report a value for the procedure. The procedure must end with the **end** command.
	Example:
	`to-report average [a b]`
	`    report (a + b) / 2`
	`end` ;; a reporter procedure that reports out the average of two numbers

`report` *value*	Immediately exits from the current to-report procedure and reports *value* as the result of that procedure. report and to-report are always used in conjunction with each other.