CS 61A Summer 2010 Week 6A Lab
Monday 7/26 Afternoon

1. The rest of the lab will familiarize you with Logo. You will build an interpreter for Logo for your third project.

To begin, type `logo` at the Unix shell prompt — **not** from Scheme! You should see something like this:

```
Welcome to Berkeley Logo version 5.5
?
```

The question mark is the Logo prompt, like the `>` in Scheme. (Later, in some of the examples below, you'll see a `>` prompt from Logo, while in the middle of defining a procedure.)

Type each of the following instruction lines and note the results. (A few of them will give error messages.) If you can't make sense of a result, ask for help.

```
print 2 + 3
```

```
print 2+3
```

```
print sum 2 3
```

```
print (sum 2 3 4 5)
```

```
print sum 2 3 4 5
```

You get the above behavior because `sum` takes a default of 2 arguments. It will take more only if you put parenthesis around it. `print` automatically takes a default of 1 argument, but if Logo detects that a procedure call follows `print`, Logo will evaluate the procedure call and give that value to `print`

```
2+3
```

```
print print 2
```

Logo doesn't automatically print return value. You need to give print instruction. And, some expressions in Logo has no output, such as `print`. It is purely an instruction.

```
print "yesterday
```

```
print "julia"
```

```
print revolution
```

```
print [blue jay way]
```

```
show [eight days a week]

show first [golden slumbers]

print first bf [she loves you]

pr first first bf [yellow submarine]
```

The brackets `[ ... ]` in Logo act like `'( ... )`.

```
to second :stuff
output first bf :stuff
end

second "something

print second "piggies

pr second [another girl]

pr first second [carry that weight]
pr second second [i dig a pony]

to pr2nd :thing
print first bf :thing
end

pr2nd [the 1 after 909]

print first pr2nd [hey jude]
```

`to fn-name :param1 :param2 ...` syntax in Logo defines a procedure. You need to end the procedure with `output ...` in order for the procedure to have a return value.

```
repeat 5 [print [this boy]]

if 3 = 1+1 [print [the fool on the hill]]

print ifelse 2=1+1
```

```
      [second [your mother should know]]
      [first "help]

print ifelse 3=1+2
      [strawberry fields forever]
      [penny lane]

print ifelse 4=1+2
      ["flying]
      [[all you need is love]]
```

`if` and `ifelse` in Logo is not a special form. The syntax is `ifelse pred [true-clause] [false-clause]` where `ifelse` would run the true-clause if pred is true, or else it will run the false-clause.

```
to greet :person
say [how are you,]
end

to say :saying
print sentence :saying :person
end

greet "ringo

show map "first [paperback writer]

show map [word first ? last ?]
         [lucy in the sky with diamonds]

to who :sent
foreach [pete roger john keith] "describe
end

to describe :person
print se :person :sent
end

who [sells out]

print :bass
```

```
make "bass "paul

print :bass

print bass

to bass
output [johnny cymbal]
end

print bass

print :bass

print "bass
```

Notice that `bass` can both be a procedure and a value. That is because logo stores names of procedures separately from names of values. `:bass` accesses the value `bass` where `bass` accesses the procedure `bass`.

```
to countdown :num
if :num=0 [print "blastoff stop]
print :num
countdown :num-1
end

countdown 5

to downup :word
print :word
if emptyp bl :word [stop]
downup bl :word
print :word
end
```