CS 61A Week 1 First Lab
Monday afternoon, Tuesday, or Wednesday morning
Try to get as much done as possible, but don't panic if you don't finish everything.

1. Start the Emacs editor, either by typing `emacs` in your main window or by selecting it from the alt-middle mouse menu. (Your TA will show you how to do this.) From the `Help` menu, select the Emacs tutorial. You need not complete the entire tutorial at the first session, but you should do so eventually.

2. Start Scheme, either by typing `stk` in your main window or by typing meta-S in your Emacs window. Type each of the following expressions into Scheme, ending the line with the Enter (carriage return) key. **Think about the results!** Try to understand how Scheme interprets what you type.

```
3
(+ 2 3)
(+ 5 6 7 8)
(+)
(sqrt 16)
(+ (* 3 4) 5)
+
'+
'hello
'(+ 2 3)
'(good morning)
(first 274)
(butfirst 274)
```

```
(first 'hello)
(first hello)
(first (bf 'hello))
(+ (first 23) (last 45))
(define pi 3.14159)
pi
'pi
(+ pi 7)
(* pi pi)
(define (square x) (* x x))
(square 5)
(square (+ 2 3))
```

3. Use Emacs to create a file called `pigl.scm` in your directory containing the Pig Latin program shown below:

```
(define (pigl wd)
  (if (pl-done? wd)
      (word wd 'ay)
      (pigl (word (bf wd) (first wd)))))

(define (pl-done? wd)
  (vowel? (first wd)))

(define (vowel? letter)
  (member? letter '(a e i o u)))
```

Make sure you are editing a file whose name ends in `.scm`, so that Emacs will know to indent your code correctly!

4. Now run Scheme. You are going to create a transcript of a session using the file you just created:

```
(transcript-on "lab1")       ; This starts the transcript file.
(load "pigl.scm")            ; This reads in the file you created earlier.
(pigl 'scheme)               ; Try out your program.
                             ; Feel free to try more test cases here!
(trace pigl)                 ; This is a debugging aid. Watch what happens
(pigl 'scheme)               ; when you run a traced procedure.
(transcript-off)
(exit)
```

5. Use `lpr` to print your transcript file.

CS 61A Week 1 Second Lab
Wednesday afternoon, Thursday, or Friday morning

1. Predict what Scheme will print in response to each of these expressions. *Then* try it and make sure your answer was correct, or if not, that you understand why!

```
(define a 3)
```

```
(define b (+ a 1))
```

```
(+ a b (* a b))
```

```
(= a b)
```

```
(if (and (> b a) (< b (* a b)))
    b
    a)
```

```
(cond ((= a 4) 6)
      ((= b 4) (+ 6 7 a))
      (else 25))
```

```
(+ 2 (if (> b a) b a))
```

```
(* (cond ((> a b) a)
         ((< a b) b)
         (else -1))
   (+ a 1))
```

```
((if (< a b) + -) a b)
```

2. In the shell, type the command

```
cp ~cs61a/lib/plural.scm .
```

(Note the period at the end of the line!) This will copy a file from the class library to your own directory. Then, using emacs to edit the file, modify the procedure so that it correctly handles cases like `(plural 'boy)`.

3. Define a procedure that takes three numbers as arguments and returns the sum of the squares of the two larger numbers.

4. Write a procedure `dupls-removed` that, given a sentence as input, returns the result of removing duplicate words from the sentence. It should work this way:

```
> (dupls-removed '(a b c a e d e b))
(c a d e b)
> (dupls-removed '(a b c))
(a b c)
> (dupls-removed '(a a a a b a a))
(b a)
```