

# SI 413 Fall 2021: Homework 3

Due Monday, September 13

Your name:

Citations and collaborators:

Comments, suggestions, or questions for your instructor:

Fill out the first row of the table on a 0-5 scale before turning in.

This rubric is also available on the website under “Admin”:

- **5:** Solution is completely correct, concisely presented, and neatly written.
- **4:** The solution is mostly correct, but one or two minor details were missed, or the presentation could be better.
- **3:** The main idea is correct, but there are some significant mistakes. The presentation is somewhat sloppy or confused.
- **2:** A complete effort was made, but the result is mostly incorrect.
- **1:** The beginning of an attempt was made, but the work is clearly incomplete.
- **0:** Not submitted.

Problem	1	2	3	4	5	6	Total
Self-assessment							
Final assessment							

Many of these exercises are programming exercises, but you do not need to submit them electronically.

## 1 Symbol Mixup

Write a function (`mixup x`) that takes an argument `x`, which can be any symbol or any number, and produces the opposite type of thing, either the number 5 if `x` is a symbol, or the symbol `'num` if `x` is a number.

For example, (`mixup 20`) should produce `'num`, and (`mixup 'hello`) should produce 5.

## 2 Nested Quotes

When you type 5 into the interpreter, it returns 5.

When you type (`quote 5`), it still returns the number 5.

But when you type (`quote (quote 5)`) or `''5`, it returns `'5`.

What do you think is going on here? Why do you need two quotes to make the symbol 5?

(Caution: this is pretty tricky. Think about how evaluation works. Play around, experiment, discuss.)

## 3 Building Blocks

In the C programming language, give an example of each of the following types of code fragments.

a) An atom (or literal)

b) A value that is not an atom

c) An expression that is not a value

d) A statement that does not end in a semicolon

## 4 Nested Lets

Write a Scheme expression that is equivalent to the following Java code, by using a series of 3 nested `let` expressions.

```
int x = 1;  
x += 3;  
x *= 12;  
return x;
```

## 5 Homoiconicity

The Wikipedia page on homoiconicity claims that raw machine code can be considered homoiconic, just like Scheme.

Explain what this means in a few sentences of your own.

Then tell me what properties of most homoiconic languages (like Scheme) does machine code definitely *not* have.

## 6 And Transformation

You know that there is a built-in function called `and` in Scheme. The built-in version actually takes any number of arguments, and always returns either `#t` or `#f`, but for this exercise we'll assume that `and` only takes two arguments.

You should be able to convince yourself that every `and` could be re-written as an `if`. For example, consider the following function that tests whether the number `x` is a “teen”.

```
(define (teen? x)
  (and (>= x 13)
       (< x 20)))
```

Well this is exactly the same as:

```
(define (teen? x)
  (if (< x 13)
      #f
      (< x 20)))
```

Your task is to write a Scheme function (`and->if expr`) that takes a quoted `and` expression and returns an equivalent quoted `if` expression that computes the same thing. (Note: the expression your code produces might not look exactly like what I have above, but it should be equivalent computationally.)

If you then `eval` the result, it should work. For example, the following should produce `#t`:

```
(define x 18)
(eval (and->if '(and (>= x 13) (< x 20))))
```