Homework 6: Scheme **hw06.zip (hw06.zip)**

Due by 11:59pm on Thursday, November 5

Instructions

Download hw06.zip (hw06.zip). Inside the archive, you will find a file called hw06.scm (hw06.scm), along with a copy of the `ok` autograder.

Submission: When you are done, submit with `python3 ok --submit`. You may submit more than once before the deadline; only the final submission will be scored. Check that you have successfully submitted your code on okpy.org (https://okpy.org/). See Lab 0 (/~cs61a/fa20/lab/lab00#submitting-the-assignment) for more instructions on submitting assignments.

Using Ok: If you have any questions about using Ok, please refer to this guide. (/~cs61a/fa20/articles/using-ok.html)

Readings: You might find the following references useful:

- Scheme Specification (/~cs61a/fa20/articles/scheme-spec.html)
- Scheme Built-in Procedure Reference (/~cs61a/fa20/articles/scheme-builtins.html)
- 3.2 (http://composingprograms.com/pages/32-functional-programming.html)

Grading: Homework is graded based on correctness. Each incorrect problem will decrease the total score by one point. There is a homework recovery policy as stated in the syllabus. **This homework is out of 2 points.**

You may find it useful to try code.cs61a.org/scheme (https://code.cs61a.org/scheme) when working through problems, as it can draw environment and box-and-pointer diagrams and it lets you walk your code step-by-step (similar to Python Tutor). Don't forget to submit your code through Ok though!

Scheme Editor

As you're writing your code, you can debug using the Scheme Editor. In your `scheme` folder you will find a new editor. To run this editor, run `python3 editor`. This should pop up a window in your browser; if it does not, please navigate to localhost:31415 (localhost:31415) and you should see it.

Make sure to run `python3 ok` in a separate tab or window so that the editor keeps running.

If you find that your code works in the online editor but not in your own interpreter, it's possible you have a bug in code from an earlier part that you'll have to track down. Every once in a while there's a bug that our tests don't catch, and if you find one you should let us know!

Required Questions

Scheme

Q1: Thane of Cadr

Define the procedures `cadr` and `caddr`, which return the second and third elements of a list, respectively:

```
(define (cddr s)
  (cdr (cdr s)))

(define (cadr s)
  'YOUR-CODE-HERE
)

(define (caddr s)
  'YOUR-CODE-HERE
)
```

Use Ok to unlock and test your code:

```
python3 ok -q cadr-caddr -u
python3 ok -q cadr-caddr
```

Hint Video

Q2: Sign

Using a `cond` expression, define a procedure `sign` that takes in one parameter `num` and returns -1 if `num` is negative, 0 if `num` is zero, and 1 if `num` is positive.

```
(define (sign num)
  'YOUR-CODE-HERE
)
```

Use Ok to unlock and test your code:

```
python3 ok -q sign -u
python3 ok -q sign
```

Hint Video

Q3: Pow

Implement a procedure `pow` for raising the number `x` to the power of a nonnegative integer `y` for which the number of operations grows logarithmically, rather than linearly (the number of recursive calls should be much smaller than the input `y`). For example, for `(pow 2 32)` should take 5 recursive calls rather than 32 recursive calls. Similarly, `(pow 2 64)` should take 6 recursive calls.

> *Hint:* Consider the following observations:
>
> 1. $b^{2k} = (b^k)^2$
> 2. $b^{2k+1} = b(b^k)^2$
>
> You may use the built-in predicates `even?` and `odd?`. Scheme doesn't support iteration in the same manner as Python, so consider another way to solve this problem.

```
(define (square x) (* x x))

(define (pow x y)
  'YOUR-CODE-HERE
)
```

Use Ok to unlock and test your code:

```
python3 ok -q pow -u
python3 ok -q pow
```

Hint Video

Submit

Make sure to submit this assignment by running:

```
python3 ok --submit
```

CS 61A (/~cs61a/fa20/)

Weekly Schedule (/~cs61a/fa20/weekly.html)

Office Hours (/~cs61a/fa20/office-hours.html)

Staff (/~cs61a/fa20/staff.html)

Resources (/~cs61a/fa20/resources.html)

Studying Guide (/~cs61a/fa20/articles/studying.html)

Debugging Guide (/~cs61a/fa20/articles/debugging.html)

Composition Guide (/~cs61a/fa20/articles/composition.html)

Policies (/~cs61a/fa20/articles/about.html)

Assignments (/~cs61a/fa20/articles/about.html#assignments)

Exams (/~cs61a/fa20/articles/about.html#exams)

Grading (/~cs61a/fa20/articles/about.html#grading)