Homework 8: Scheme Lists

hw08.zip (hw08.zip)

Due by 11:59pm on Thursday, April 11

Instructions

Download hw08.zip (hw08.zip). Inside the archive, you will find a file called hw08.scm (hw08.scm), along with a copy of the `ok` autograder.

Submission: When you are done, submit the assignment by uploading all code files you've edited to Gradescope. 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 Gradescope. See Lab 0 (/~cs61a/sp24/lab/lab00#task-c-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/sp24/articles/using-ok)

Readings: You might find the following references useful:

- Scheme Specification (/~cs61a/sp24/articles/scheme-spec/)
- Scheme Built-in Procedure Reference (/~cs61a/sp24/articles/scheme-builtins/)

Grading: Homework is graded based on correctness. Each incorrect problem will decrease the total score by one point. **This homework is out of 2 points.**

The 61A Scheme interpreter is included in each Scheme assignment. To start it, type `python3 scheme` in a terminal. To load a Scheme file called `f.scm`, type `python3 scheme -i f.scm`. To exit the Scheme interpreter, type `(exit)`.

Scheme Editor

All Scheme assignments include a web-based editor that makes it easy to run ok tests and visualize environments. Type `python3 editor` in a terminal, and the editor will open in a browser window (at `http://127.0.0.1:31415/`). To stop running the editor and return to the command line, type `Ctrl-C` in the terminal where you started the editor.

The `Run` button loads the current assignment's `.scm` file and opens a Scheme interpreter, allowing you to try evaluating different Scheme expressions.

The `Test` button runs all ok tests for the assignment. Click `View Case` for a failed test, then click `Debug` to step through its evaluation.

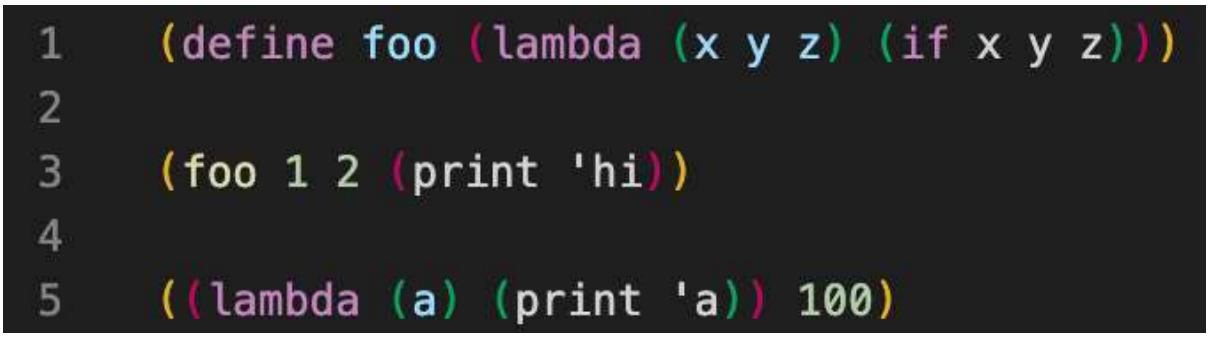
Recommended VS Code Extensions

If you choose to use VS Code as your text editor (instead of the web-based editor), install the vscode-scheme (https://marketplace.visualstudio.com/items?itemName=sjhuangx.vscode-scheme) extension so that parentheses are highlighted.

Before:

```
1    (define foo (lambda (x y z) (if x y z)))
2
3    (foo 1 2 (print 'hi))
4
5    ((lambda (a) (print 'a)) 100)
```

After:



In addition, the 61a-bot (installation instructions (/~cs61a/sp24/articles/61a-bot)) VS Code extension is available for Scheme homeworks. The bot is also integrated into `ok`.

Required Questions

Required Questions

Getting Started Videos

Q1: Ascending

Implement a procedure called `ascending?`, which takes a list of numbers `s` and returns `True` if the numbers are in non-descending order, and `False` otherwise.

A list of numbers is non-descending if each element after the first is greater than or equal to the previous element. For example...

- `(1 2 3 3 4)` is non-descending.
- `(1 2 3 3 2)` is not.

> **Hint**: The built-in `null?` procedure returns whether its argument is `nil`.

> **Note**: The question mark in `ascending?` is just part of the procedure name and has no special meaning in terms of Scheme syntax. It is a common practice in Scheme to name procedures with a question mark at the end if it returns a boolean value.

```
(define (ascending? s)
  'YOUR-CODE-HERE
)
```

Use Ok to unlock and test your code:

```
python3 ok -q ascending -u
python3 ok -q ascending
```

Q2: My Filter

Write a procedure `my-filter`, which takes a predicate `pred` and a list `s`, and returns a new list containing only elements of the list that satisfy the predicate. The output should contain the elements in the same order that they appeared in the original list.

Note: Make sure that you are not just calling the built-in `filter` function in Scheme - we are asking you to re-implement this!

```
(define (my-filter pred s)
  'YOUR-CODE-HERE
)
```

Use Ok to unlock and test your code:

```
python3 ok -q filter -u
python3 ok -q filter
```

Q3: Interleave

Implement the function `interleave`, which takes two lists `lst1` and `lst2` as arguments. `interleave` should return a new list that interleaves the elements of the two lists. (In other words, the resulting list should contain elements alternating between `lst1` and `lst2`, starting at `lst1`).

If one of the input lists to `interleave` is shorter than the other, then `interleave` should alternate elements from both lists until one list has no more elements, and then the remaining elements from the longer list should be added to the end of the new list.

```
(define (interleave lst1 lst2)
  'YOUR-CODE-HERE
)
```

Use Ok to unlock and test your code:

```
python3 ok -q interleave -u
python3 ok -q interleave
```

Q4: No Repeats

Implement `no-repeats`, which takes a list of numbers `s`. It returns a list that has all of the unique elements of `s` in the order that they first appear, but no repeats.

For example, `(no-repeats (list 5 4 5 4 2 2))` evaluates to `(5 4 2)`.

> **Hint:** You may find it helpful to use `filter` with a `lambda` procedure to filter out repeats. To test if two numbers `a` and `b` are not equal, use `(not (= a b))`.

```
(define (no-repeats s)
  'YOUR-CODE-HERE
)
```

Use Ok to test your code:

```
python3 ok -q no_repeats
```

Submit

Submit this assignment by uploading any files you've edited **to the appropriate Gradescope assignment.** Lab 00 (https://cs61a.org/lab/lab00/#submit-with-gradescope) has detailed instructions.

In addition, all students who are **not** in the mega lab must complete this attendance form (https://go.cs61a.org/lab-att). Submit this form each week, whether you attend lab or missed it for a good reason. The attendance form is not required for mega section students.

Exam Practice

The following are some Scheme List exam problems from previous semesters that you may find useful as additional exam practice.

1. Fall 2022 Final, Question 8: A Parentheses Scheme (https://cs61a.org/exam/fa22/final/61a-fa22-final.pdf#page=20)
2. Spring 2022 Final, Question 11: Beadazzled, The Scheme-quel (https://cs61a.org/exam/sp22/final/61a-sp22-final.pdf#page=23)
3. Fall 2021 Final, Question 4: Spice (https://cs61a.org/exam/fa21/final/61a-fa21-final.pdf#page=18)