Homework 9: Interpreters, Macros

hw09.zip (hw09.zip)

Due by 11:59pm on Tuesday, November 29

Instructions

Download hw09.zip (hw09.zip). Inside the archive, you will find a file called hw09.scm (hw09.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 (/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. (/articles/using-ok)

Readings: You might find the following references useful:

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

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.**

Macros are a method of programming that allow programmers to treat expressions as data and create procedures of a language using the language itself. Macros open the door to many clever tricks of creating "shortcuts", and with Scheme, allow us to build our own special forms other than the ones that are built in.

Required Questions

Getting Started Videos

Interpreters

Q1: WWSD: Eval and Apply

How many calls to `scheme_eval` and `scheme_apply` would it take to evaluate each of these Scheme expressions?

> You may find the Interpreters Study Guide (https://cs61a.org/study-guide/interpreters/#counting-calls) helpful.
>
> Use Ok to test your knowledge by writing the number of calls needed to evaluate each expression:
>
> ```
> python3 ok -q wwsd-eval_apply -u
> ```

```
scm> (+ 2 4 6 8) ; number of calls to scheme_eval
------

scm> (+ 2 4 6 8) ; number of calls to scheme_apply
------

scm> (+ 2 (* 4 (- 6 8))) ; number of calls to scheme_eval
------

scm> (+ 2 (* 4 (- 6 8))) ; number of calls to scheme_apply
------

scm> (if #f (+ 2 3) (+ 1 2)) ; number of calls to scheme_eval
------

scm> (if #f (+ 2 3) (+ 1 2)) ; number of calls to scheme_apply
------

scm> (define (cube a) (* a a a)) ; number of calls to scheme_eval
------

scm> (define (cube a) (* a a a)) ; number of calls to scheme_apply
------

scm> (cube 3) ; number of calls to scheme_eval
------

scm> (cube 3) ; number of calls to scheme_apply
------
```

Macros

Q2: When Macro

Using macros, define a new special form, `when`, that has the following structure:

```
(when <condition>
      (<expr1> <expr2> <expr3> ...))
```

If the condition is not false (a truthy expression), all the subsequent operands are evaluated in order and the value of the last expression is returned. Otherwise, the entire `when` expression evaluates to `okay`.

> Hint: you may find the begin (https://cs61a.org/articles/scheme-spec/#begin) form useful.

```
scm> (when (= 1 0) ((/ 1 0) 'error))
okay

scm> (when (= 1 1) ((print 6) (print 1) 'a))
6
1
a
```

```
(define-macro (when condition exprs)
  'YOUR-CODE-HERE
  )
```

Use Ok to test your code:

```
python3 ok -q when-macro
```

Q3: Switch

Define the macro `switch`, which takes in an expression `expr` and a list of pairs, `cases`, where the first element of the pair is some *value* and the second element is a single expression. `switch` will evaluate the expression contained in the list of `cases` that corresponds to the value that `expr` evaluates to.

```
scm> (switch (+ 1 1) ((1 (print 'a))
                      (2 (print 'b))
                      (3 (print 'c))))
b
```

You may assume that the value `expr` evaluates to is always the first element of one of the pairs in `cases`. You can also assume that the first value of each pair in `cases` is a value.

```
(define-macro (switch expr cases)
        (cons _________
              (map (_________ (_________) (cons _________ (cdr case)))
                      cases))
)
```

Use Ok to test your code:

```
python3 ok -q switch
```

Exam Practice

Homework assignments will also contain prior exam questions for you to try. These questions have no submission component; feel free to attempt them if you'd like some practice!

Macros

1. Fall 2019 Final Q9: Macro Lens (https://cs61a.org/exam/fa19/final/61a-fa19-final.pdf#page=10)
2. Summer 2019 Final Q10c: Slice (https://cs61a.org/exam/su19/final/61a-su19-final.pdf#page=10)
3. Spring 2019 Final Q8: Macros (https://cs61a.org/exam/sp19/final/61a-sp19-final.pdf#page=8)