CSCI 275: Programming Abstractions Exam 2 Review Spring 2025

Stephen Checkoway Slides from Molly Q Feldman



Plan for Today

- Brief overview of Exam 2 logistics
- Many (!) Clicker questions for Review
- Some open ended Review Questions
- If there's time, opportunity to ask questions

Details of the Exam Open book/notes/Racket – will specify specifically in the assignment

- programming
- Two goals:
- Show you how much you know
- Show me your "intuitive" response to different topics

Programming problems & conceptual questions related to

Logistics

Exam will be available for 24 hours starting at 1:30 p.m. on Monday

The exam will be released/submitted via GitHub Classroom

During Monday's class, I will be doing my jury duty

On Tuesday, I'll be in my office from 11–12 and available to answer questions

Monday

You can also ask private questions on Ed, but I'll be busy all day

SAME deal as Exam 1



In Scope Topics

- Higher-order functions
- MiniScheme
 - Implementation
 - Design
 - Theory
- Scoping
- <u>Streams</u>
- Parameter Passing

Practice Clicker Questions

Consider a new structure to represent a point in 2D: (struct point (x y) #:transparent)

If p is a point created via the point constructor, how would we create a new point whose fields are the absolute value of the fields in p? (The function (abs x) returns the absolute value of x.)

- A. (map abs p)
- B. (list* 'point (map abs (rest p)))
- D. (point (abs (point-x p)) (abs (point-y p)))
- E. More than one of the above (which?)

C. (struct point (abs (point-x p)) (abs (point-y p)))



Which of the following, when (stream->list (stream-take (PROCEDURE 1 2) 10)) is run, produces ' (1 2 1 2 1 2 1 2 1 2)? (define (sheep a b) Α. (stream-cons '(a b) (sheep a b)))

(define (lamb a b) B. (stream-cons a (stream-cons b (lamb a b))))

C. (define (ram a b) (stream-cons a b (ram a b)))

D. More than one of the above E. None of the above



When parsing a let expression, which pieces of information does the parse tree need to store?

- A. An extended environment mapping the symbols in the binding list to their values and the body expression
- B. A list of binding symbols, list of parse trees for the binding expressions, and the parsed body expression
- C. A list of binding symbols, a list of binding values, and the body expression
- D. Any of A, B, or C work
- E. Either B or C work, but not A

would need to change?

- (let* ([x 2] [y (+ x 4)]) y)
 - A.env.rkt, parse.rkt, interp.rkt
 - B.interp.rkt and parse.rkt
 - C.parse.rkt only
 - D.interp.rkt only
 - E. Some other combination!

Let's say we want to implement let* in MiniScheme. Which files



Evaluating a lambda gives a closure. A closure in a language with *dynamic binding* needs to contain which information?

- A. The list of parameters
- B. The list of parameters and the parsed body
- C. The list of parameters, the parsed body, and the environment in which the lambda was evaluated
- D. The list of parameters, the parsed body, and the environment in which the closure is to be evaluated

Additional Practice



In our implementation of primitive procedures in MiniScheme, would something like the following work to handle the lt? procedure? (define (apply-primitive-op op args) (cond ...

- [(eq? op 'lt?) (apply < args)]</pre> ...))
- A. It will work because < is Racket's less than
- number of arguments
- C. It won't work because < returns #t or #f
- D. It won't work because the arguments might not be numbers
- E. More than one of the above (which?)

B. It won't work because < takes two arguments and apply allows any

What is the value of the express about dynamic binding?

What is the value of the expression assuming lexical binding? What

A. Lexical: 100 Dynamic: 100

C. Lexical: 200 Dynamic: 100

D. Lexical: 200 Dynamic: 200

E. Lexical: 200 Dynamic: 400

containing the powers of n.

For instance, if n = 2, we should get the stream (2,4,8,16,32...).

Write a procedure power that, given n, returns a stream

Why do we have multiple environments?

bindings for each let expression or procedure call?

Why not just have a single environment where we update the