Part 1 Summary

Due November 8th at 10pm

*** indicates an extra credit function

ABSTRACTION: binding

(make-binding var val) constructor : create a new binding with var bound to val

(binding-variable binding) selector : returns variable of binding

(binding-value binding) selector : returns value of binding

ABSTRACTION: frame

(make-frame vars vals) constructor : create a new frame with vars bound to vals

(empty-frame? frame) tester : returns #t if frame is empty; #f otherwise

(first-binding frame) selector : returns the first binding in frame

(rest-of-bindings frame) selector : returns the rest of the bindings in frame

(adjoin-binding binding frame) creates a new frame with binding added to frame

(binding-in-frame var frame) returns binding for var; #f if no binding exists

ABSTRACTION: environment

(empty-env) representation of an empty environment

(empty-env? env) tester : returns #t if env is empty; #f otherwise

(first-frame env) selector : returns the first frame in env

(rest-of-frames env) selector : returns the rest of the frames in env

(set-first-frame! env new-frame) mutator : replace first frame of env with new-frame

(adjoin-frame frame env) creates a new environment with frame added to env

(extend-env vars vals env) calls adjoin-frame with a new frame (whose vars are bound to vals) and env

(binding-in-env var env) returns the binding for var in env; #f if no binding exists

(setup-env) returns a new global environment

global-env stores the global environment

IMPLEMENTATION: variables

(lookup-variable var env) returns the value of var in env; error otherwise

(variable? exp) tester : returns #t if exp is a variable; #f otherwise

HELPER: tagged-list?

(tagged-list? exp tag) tester : returns #t if exp is a list beginning with tag; #f otherwise

***tagged-list-length-n? exp tag n)** tester : returns #t if exp is a list of length n beginning with tag; #f otherwise

***tagged-list-min-length-n? exp tag n)** tester : returns #t if exp is a list of length n or longer beginning with tag; #f otherwise

IMPLEMENTATION: quote

(quoted? exp) tester : returns #t if exp is a quoted expression; #f otherwise

(text-of-quotation exp) returns the text of a quoted expression

IMPLEMENTATION: define

(definition? exp) tester : returns #t if exp is a definition; #f otherwise

(eval-definition exp env) controller : define variable as specified in exp

(definition-variable exp) selector : returns variable portion of define exp

(definition-value exp) selector : returns value portion of define exp

(define-variable! var val env) add a binding for var bound to val in the first frame of env (or print an error if a binding already exists)

IMPLEMENTATION: evaluation

(i-eval exp env) controller : evaluates exp in env

(read-eval-print-loop) controller : read an expression from the user; evaluate; print result; loop