Part 2 Summary
Due November 15th at 10pm

*** indicates an extra credit function

IMPLEMENTATION: application

(i-apply proc args) controller : evaluates procedures proc with arguments args
(application? exp) tester : returns #t if exp is a procedure application; #f otherwise
(operator exp) selector : returns operator (proc) of application
(operands exp) selector : returns operands (args) of application
(eval-operands operands env) returns a list of evaluated operands
(eval-application exp env) evaluates a procedure application

ABSTRACTION: primitive

(make-primitive name proc) constructor : creates a new primitive
(primitive-procedure? proc) tester : returns #t if proc is a primitive; #f otherwise
(primitive-name proc) selector : returns name of primitive procedure
(primitive-implementation proc) selector : returns implementation of primitive procedure
(apply-primitive-procedure proc vals) controller : apply primitive proc to evaluated arguments vals

IMPLEMENTATION: begin

(begin? exp) tester : returns #t if exp is a begin expression; #f otherwise
(begin-expressions exp) selector : returns list of expressions in begin exp
(eval-begin exp env) controller : evaluate begin expression

IMPLEMENTATION: if

(if? exp) tester : returns #t if exp is an if expression; #f otherwise
(test-expression exp) selector : returns test expression of an if expression
(then-expression exp) selector : returns then expression of an if expression
(else-expression exp) selector : returns else expression of an if expression
(eval-if exp env) controller : evaluate if expression
IMPLEMENTATION: cond

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

(first-cond-exp exp) selector: returns first conditional in cond exp

(rest-of-cond-exps exp) selector: returns remaining conditionals in cond expression

(eval-cond exp env) controller: evaluate cond expression

IMPLEMENTATION: and

*** (and? exp) testor: returns #t if exp is an and expression; #f otherwise

*** (eval-and exp env) controller: evaluate and expression

IMPLEMENTATION: or

*** (or? exp) testor: returns #t if exp is an or expression; #f otherwise

*** (eval-or exp env) controller: evaluate or expression