1 Interpreter

1.1 Additional Preliminaries

Save your solutions for this problem to a file named eval.rkt.

Run the following expression (you must have required [extras.rkt](http://www.ccs.neu.edu/course/cs5010sp15/files/extras.rkt)) to check that your file is properly named and is in the proper directory:

(check-location "09" "eval.rkt")

Add these additional provides at the top of your file (below the requires), so that we can test your solution:

([provide](http://docs.racket-lang.org/reference/require.html#%28form._%28%28lib._racket%2Fprivate%2Fbase..rkt%29._provide%29%29) eval)

([provide](http://docs.racket-lang.org/reference/require.html#%28form._%28%28lib._racket%2Fprivate%2Fbase..rkt%29._provide%29%29) lambda?)

([provide](http://docs.racket-lang.org/reference/require.html#%28form._%28%28lib._racket%2Fprivate%2Fbase..rkt%29._provide%29%29) errstr?)

([provide](http://docs.racket-lang.org/reference/require.html#%28form._%28%28lib._racket%2Fprivate%2Fbase..rkt%29._provide%29%29) subst)

([provide](http://docs.racket-lang.org/reference/require.html#%28form._%28%28lib._racket%2Fprivate%2Fbase..rkt%29._provide%29%29) expr->expr/no-var)

([provide](http://docs.racket-lang.org/reference/require.html#%28form._%28%28lib._racket%2Fprivate%2Fbase..rkt%29._provide%29%29) expr=?)

1.2 Problem Description

Although you are having a great time programming with Racket, you’ve decided to create your own programming language, called PDPLang . Your task for this assignment is to implement an interpreter for this language.

Note: Though this assignment is self-contained, you may find it helpful by reviewing[Chapter 24](http://www.ccs.neu.edu/home/matthias/HtDP2e/Draft/part_4.html#%28part._ch~3aevaluator%29) of the textbook.

Specifically, design the following function:

|  |
| --- |
| ; eval : Program -> ListOf<Result> |
| ; Evaluates a PDPLang program to a list of Results. |
| ; Specifically, evaluates the Exprs in p, in the context of the given Defs. |
| ; WHERE: A function may be called before it is defined. |
| ; WHERE: The produced results are in the same relative order as their |
| ; originating Exprs in p. |
| ([define](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._define%29%29) (eval p) [...](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._......%29%29)) |

To help us test, please also implement and provide the following predicates:

|  |
| --- |
| ; lambda? : Expr -> Boolean |
| ; Returns true if e is a PDPLang lambda expression. |
| ([define](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._define%29%29) (lambda? e) [...](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._......%29%29)) |
| ; errstr? : Expr -> Boolean |
| ; Returns true if e is a PDPLang ErrString expression. |
| ([define](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._define%29%29) (errstr? e) [...](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._......%29%29)) |

Assume that PDPLang programs (i.e., [information](http://www.ccs.neu.edu/course/cs5010sp15/recipe.html#%28tech._information%29)) have already been parsed to the following *data* representations.

|  |
| --- |
| ; A UniqueListOf<X> is a ListOf<X> |
| ; WHERE: none of the elements of the list are equal? to each other |
|  |
| ; A 2ListOf<X> is a (cons X (cons X ListOf<X>)) |
| ; Represents a list of 2 or more elements. |
|  |
| ; A Program is a: |
| ; - empty |
| ; - (cons Expr Program) |
| ; - (cons Def Program) |
| ; Represents a PDPLang program consisting of function defs and expressions. |
| ; WHERE: no two Defs have the same name |
|  |
| ; An Expr is one of: |
| ; - Number |
| ; - Boolean |
| ; - Var |
| ; - ErrString |
| ; - Lambda |
| ; - (make-arith ArithOp 2ListOf<Expr>) ; an arithmetic expression |
| ; - (make-bool BoolOp 2ListOf<Expr>)   ; a boolean expression |
| ; - (make-cmp CmpOp 2ListOf<Expr>)     ; a comparison expression |
| ; - (make-if-exp Expr Expr Expr) ; an if conditional |
| ; - (make-call Expr ListOf<Expr>) ; a function call |
| ; Represents a PDPLang expression. |
| ([define-struct](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._define-struct%29%29) arith (op args)) |
| ([define-struct](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._define-struct%29%29) bool (op args)) |
| ([define-struct](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._define-struct%29%29) cmp (op args)) |
| ([define-struct](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._define-struct%29%29) if-exp (test branch1 branch2)) |
| ([define-struct](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._define-struct%29%29) call (fn args)) |
|  |
| ; A Var is a Symbol, representing PDPLang variable. |
|  |
| ; An ErrString is a String, representing a PDPLang error message. |
|  |
| ; A Lambda is a (make-lam UniqueListOf<Param> Expr) |
| ; Represents a lambda expression in PDPLang |
| ([define-struct](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._define-struct%29%29) lam (params body)) |
|  |
| ; A Param is a Var, representing a function parameter. |
|  |
| ; An ArithOp is one of: |
| ; - '+ |
| ; - '- |
| ; - '\* |
| ; - '/ |
| ; Represents an arithmetic operation in PDPLang |
|  |
| ; A BoolOp is one of: |
| ; - 'and |
| ; - 'or |
| ; Represents a boolean operation in PDPLang |
|  |
| ; A CmpOp is one of: |
| ; - '= |
| ; - '< |
| ; - '> |
| ; Represents a comparison operation in PDPLang |
|  |
| ; A Def is a (make-def FnName UniqueListOf<Param> Expr) |
| ; Represents the definition of a function with the specified parameters. |
| ([define-struct](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._define-struct%29%29) def (name params body)) |
|  |
| ; A FnName is a Var, representing a function name. |
|  |
| ; A Result is a: |
| ; - Number |
| ; - Boolean |
| ; - ErrString |
| ; - Lambda |

Complete the [Data Design](http://www.ccs.neu.edu/course/cs5010sp15/recipe.html#%28part._data%29) by adding templates and data examples.

Here’s how to evaluate a PDPLang Program.

* A PDPLang Program evaluates to a list of Results, where the list of Results are result of evaluating the Exprs in the program.
* Exprs in a PDPLang Program are evaluated assuming that all the Defs in the program are already defined.

Here’s how to evaluate a PDPLang Expr.

* An Expr that is already a Result evaluates to itself.
* A Var, if it represents a defined Def, should evaluate to an equivalent lambda Expr. Otherwise, a Var should evaluate to an appropriate ErrStr.

|  |
| --- |
| ([define](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._define%29%29) F (make-def 'f '(x) 'x)) |
| ([check-pred](http://docs.racket-lang.org/rackunit/api.html#%28def._%28%28lib._rackunit%2Fmain..rkt%29._check-pred%29%29) lambda? ([first](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28def._htdp-intermediate-lambda._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._first%29%29) (eval ([list](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28def._htdp-intermediate-lambda._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._list%29%29) F 'f))) "result is lambda") |
| ([check-pred](http://docs.racket-lang.org/rackunit/api.html#%28def._%28%28lib._rackunit%2Fmain..rkt%29._check-pred%29%29) errstr? ([first](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28def._htdp-intermediate-lambda._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._first%29%29) (eval ([list](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28def._htdp-intermediate-lambda._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._list%29%29) F 'g))) "err: undefined var") |
| ([check-pred](http://docs.racket-lang.org/rackunit/api.html#%28def._%28%28lib._rackunit%2Fmain..rkt%29._check-pred%29%29) errstr? ([first](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28def._htdp-intermediate-lambda._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._first%29%29) (eval ([list](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28def._htdp-intermediate-lambda._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._list%29%29) 'x))) "err: undefined var") |

* Arithmetic and boolean operations evaluate to the obvious Result.

UPDATE 2015-03-19: Boolean operations are not short circuiting.

* An if expression:
  + first evaluates its test expression.
  + The result of evaluating an if expression is the result of evaluating the first branch if the test expression evaluates to [true](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28def._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._true%29%29).
  + The result of evaluating an if expression is the result of evaluating the second branch if the test expression evaluates to [false](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28def._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._false%29%29).
  + Otherwise, the result of evaluating the if expression should be an appropriate error message.
* Evaluate a function call by:
  + evaluating its function subexpression; if the result is not a Lambda, the result of the function call is an appropriate ErrString;
  + evaluating its arguments;
  + replacing each parameter reference in the function body with its corresponding argument and evaluating the resulting expressions; a function call evaluates to an appropriate ErrString if the number of arguments does not match the number of function parameters.
  + To help you with function call evaluation, implement, provide and use the following function:

|  |
| --- |
| ; subst : Result Var Expr -> Expr |
| ; Replaces references to x in e with r. |
| ; Does not replace x with r if x occurs in the body of a lambda |
| ; that shadows x. |
| ; WHERE: r has no unbound variables |
| (begin-for-test |
| ([check-equal?](http://docs.racket-lang.org/rackunit/api.html#%28def._%28%28lib._rackunit%2Fmain..rkt%29._check-equal~3f%29%29) (subst 4 'x 'x) 4 "x matches") |
| ([check-equal?](http://docs.racket-lang.org/rackunit/api.html#%28def._%28%28lib._rackunit%2Fmain..rkt%29._check-equal~3f%29%29) (subst 4 'y 'x) 'x "y doesnt match") |
| ([check-equal?](http://docs.racket-lang.org/rackunit/api.html#%28def._%28%28lib._rackunit%2Fmain..rkt%29._check-equal~3f%29%29) |
| (subst 4 'x (make-arith '+ '(x 5))) |
| (make-arith '+ '(4 5)) |
| "subst in arith") |
| ([check-equal?](http://docs.racket-lang.org/rackunit/api.html#%28def._%28%28lib._rackunit%2Fmain..rkt%29._check-equal~3f%29%29) |
| (subst 4 'x (make-lam '(y) (make-arith '+ '(x y)))) |
| (make-lam '(y) (make-arith '+ '(4 y))) |
| "subst in lambda") |
| ([check-equal?](http://docs.racket-lang.org/rackunit/api.html#%28def._%28%28lib._rackunit%2Fmain..rkt%29._check-equal~3f%29%29) |
| (subst 4 'x (make-lam '(x) (make-arith '+ '(x 5)))) |
| (make-lam '(x) (make-arith '+ '(x 5))) |
| "dont subst shadowed vars in lambdas")) |
| ([define](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._define%29%29) (subst r x e) [...](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._......%29%29)) |

* + UPDATE 2015-03-20: Here are some function call examples:

|  |
| --- |
| ([define](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._define%29%29) MK-ADD-DEF |
| (make-def |
| 'mk-add '(n) |
| (make-lam '(m) (make-arith '+ '(n m))))) |
| ([define](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._define%29%29) ADD5-DEF |
| (make-def |
| 'add5 '(x) |
| (make-call (make-call 'mk-add '(5)) '(x)))) |
| ; add5-or-6 : adds 5 to y if y it's positive, else adds 6 |
| ([define](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._define%29%29) ADD5-OR-6-DEF |
| (make-def |
| 'add5-or-6 '(y) |
| (make-if-exp (make-cmp '> '(y 0)) |
| (make-call 'add5 '(y)) |
| (make-call (make-call 'mk-add '(6)) '(y))))) |
| ([check-equal?](http://docs.racket-lang.org/rackunit/api.html#%28def._%28%28lib._rackunit%2Fmain..rkt%29._check-equal~3f%29%29) |
| (eval |
| ([list](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28def._htdp-intermediate-lambda._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._list%29%29) |
| MK-ADD-DEF |
| ADD5-DEF |
| ADD5-OR-6-DEF |
| (make-call 'add5 '(10)) |
| (make-call 'add5-or-6 '(200)) |
| (make-call 'add5-or-6 '(-100)))) |
| ([list](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28def._htdp-intermediate-lambda._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._list%29%29) 15 205 -94) |
| "call-fn evaluates to a function") |

* Any expressions whose subexpressions evaluate to the wrong kind of Result should evaluate to an ErrString.

|  |
| --- |
| ([check-true](http://docs.racket-lang.org/rackunit/api.html#%28def._%28%28lib._rackunit%2Fmain..rkt%29._check-true%29%29) |
| ([ormap](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html" \l "%28def._htdp-intermediate-lambda._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._ormap%29%29) |
| errstr? |
| (eval ([list](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28def._htdp-intermediate-lambda._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._list%29%29) (make-arith '+ ([list](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28def._htdp-intermediate-lambda._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._list%29%29) 2 [true](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28def._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._true%29%29))) |
| (make-bool 'and ([list](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28def._htdp-intermediate-lambda._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._list%29%29) 2 [true](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28def._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._true%29%29))) |
| (make-if-exp 3 1 2)))) |
| "wrong kind of args") |

UPDATE 2015-03-18: Finally, implement expr->expr/no-var and expr=? (this part of problem set is based on [exercise 451](http://www.ccs.neu.edu/home/matthias/HtDP2e/Draft/part_six.html#%28counter._%28figure._fig~3asd%29%29) of the textbook).

You might also want to review the [in-class exercise from Module07](http://www.ccs.neu.edu/course/cs5010sp15/module07.html#%28part._mod07~3aclass%29) that computes tree depths.

To help you even more, here are slides from a previous semester for somewhat-related problem:

* Accumulators and Depth [[pptx](http://www.ccs.neu.edu/course/cs5010sp15/files/Lesson%207.1%20Accumulators.pptx)] [[pdf](http://www.ccs.neu.edu/course/cs5010sp15/files/Lesson%207.1%20Accumulators.pdf)]
* Free Variables [[pptx](http://www.ccs.neu.edu/course/cs5010sp15/files/Lesson%207.3%20Free%20Variables%20Case%20Study.pptx)] [[pdf](http://www.ccs.neu.edu/course/cs5010sp15/files/Lesson%207.3%20Free%20Variables%20Case%20Study.pdf)]

|  |
| --- |
| ; An *ExprNoVar* is one of: |
| ; - Number |
| ; - Boolean |
| ; - [StaticDist](http://www.ccs.neu.edu/course/cs5010sp15/set09.html#%28tech._staticdist%29) |
| ; - ErrString |
| ; - [LamNoVar](http://www.ccs.neu.edu/course/cs5010sp15/set09.html#%28tech._lamnovar%29) |
| ; - (make-arith ArithOp 2ListOf<ExprNoVar>) ; an arithmetic expression |
| ; - (make-bool BoolOp 2ListOf<ExprNoVar>)   ; a boolean expression |
| ; - (make-cmp CmpOp 2ListOf<ExprNoVar>)     ; a comparison expression |
| ; - (make-if-exp ExprNoVar ExprNoVar ExprNoVar) ; an if conditional |
| ; - (make-call ExprNoVar ListOf<ExprNoVar>) ; a function call |
| ; Represents an Expr without explicit variables. |
|  |
| ; A *StaticDist* is a (list Depth Index) |
| ; Represents a variable reference |
| ; where depth is number of additional lambdas between this var ref and the |
| ; lambda for which this variable is a parameter, |
| ; and index is the (0-based) position of this variable in that lambda's |
| ; parameter list. |
|  |
| ; A Depth is a Natural |
| ; An Index is a Natural |
|  |
| ; A *LamNoVar* is a (make-lam/no-var ExprNoVar) |
| ([define-struct](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._define-struct%29%29) lam/no-var (body)) |
|  |
| ; expr->expr/no-var : Expr -> ExprNoVar |
| ; Replaces Var in e with StaticDist. |
| ; WHERE: there are no unbound variables in e. |
| (begin-for-test |
| ([check-equal?](http://docs.racket-lang.org/rackunit/api.html#%28def._%28%28lib._rackunit%2Fmain..rkt%29._check-equal~3f%29%29) |
| (expr->expr/no-var (make-lam '(x) 'x)) |
| (make-lam/no-var '(0 0)) |
| "basic lambda") |
| ([check-equal?](http://docs.racket-lang.org/rackunit/api.html#%28def._%28%28lib._rackunit%2Fmain..rkt%29._check-equal~3f%29%29) |
| (expr->expr/no-var (make-lam '(x y) (make-lam '(z) (make-call 'x '(y z))))) |
| (make-lam/no-var (make-lam/no-var (make-call '(1 0) '((1 1) (0 0))))) |
| "nested lambdas")) |
| ([define](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._define%29%29) (expr->expr/no-var e) [...](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._......%29%29)) |
|  |
| ; expr=? : Expr Expr -> Boolean |
| ; Returns true if e1 and e2 are structurally equivalent, up to some |
| ; renaming of variable names. |
| (begin-for-test |
| ([check](http://docs.racket-lang.org/rackunit/api.html#%28def._%28%28lib._rackunit%2Fmain..rkt%29._check%29%29) |
| expr=? |
| (make-lam '(x) 'x) |
| (make-lam '(y) 'y) |
| "equivalent basic lambdas") |
| ([check-false](http://docs.racket-lang.org/rackunit/api.html#%28def._%28%28lib._rackunit%2Fmain..rkt%29._check-false%29%29) |
| (expr=? |
| (make-lam '(x y) (make-call 'x '(y))) |
| (make-lam '(y x) (make-call 'x '(y)))) |
| "not equivalent") |
| ([check](http://docs.racket-lang.org/rackunit/api.html#%28def._%28%28lib._rackunit%2Fmain..rkt%29._check%29%29) |
| expr=? |
| (make-lam '(y) (make-lam '(x) (make-call 'y '(x)))) |
| (make-lam '(x) (make-lam '(y) (make-call 'x '(y)))) |
| "equivalent nested-lambdas")) |
| ([define](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._define%29%29) (expr=? e1 e2) [...](http://docs.racket-lang.org/htdp-langs/intermediate-lam.html#%28form._%28%28lib._lang%2Fhtdp-intermediate-lambda..rkt%29._......%29%29)) |