

CSCI.4430/6430 Programming Languages Spring 2015
Programming Assignment #2

*This assignment is to be done either **individually** or **in pairs**. Do not show your code to any other group and do not look at any other group's code. Do not put your code in a public directory or otherwise make it public. However, you may get help from the TAs or the instructor. You are encouraged to use the LMS Discussions page to post problems so that other students can also answer/see the answers.*

Lambda Calculus Interpreter

The goal of this assignment is to write a lambda calculus interpreter in Oz to reduce lambda expressions in a call-by-value (applicative order) manner.

You are to use the following grammar for the lambda calculus:

```

<expression> ::= <atom>
                | lambda(<atom> <expression>)
                | [<expression> <expression>]
    
```

Your interpreter is expected to take the lambda calculus expression and perform repeatedly beta reduction until no longer possible (a value expression that can no longer be beta-reduced) and then eta reduction until no longer possible.

Once your interpreter is finished, you should be able to execute statements such as:

```
{Browse {Run [lambda(x x) y]}} % should display y
```

Below are more test cases:

Expression	Result	Comment
[lambda(x lambda(y [x y])) [y w]]	lambda(z [[y w] z])	Avoid capturing the free variable y in [y w]
[lambda(x x) y]	y	Identity combinator
lambda(x [y x])	y	Eta reduction
[[lambda(y lambda(x [y x])) lambda(x [x x])] y]	[y y]	Application combinator
[[[lambda(b lambda(t lambda(e [[b t] e]))) lambda(x lambda(y x))] x] y]	x	If-then-else combinator
lambda(x [[lambda(x [y x]) lambda(x [z x])] x])	[y z]	Eta reductions

```
[lambda(y [lambda(x lambda(y [x y]))
y]) [y w]]
```

```
[y w]
```

Alpha renaming, beta reduction
and eta reduction all involved

Hints: You may define auxiliary procedures for alpha-renaming, beta-reduction, and eta-reduction. For beta reduction, you may want to write an auxiliary procedure that substitutes all occurrences of a variable in an expression for another expression. Be sure that the replacing expression does not include free variables that would become captured in the substitution. Remember that in call-by-value, the argument to a function is evaluated before the function is called.

Extra Credit (Up to 25% bonus)

1. Create a call-by-name (normal order evaluation) lambda expression interpreter. Your interpreter should be able to bypass the infinite loop during the reduction of the following expression:

```
[lambda(x y) [lambda(x [x x]) lambda(x [x x])]] % should display y
```

2. Write a function `{AreEqual E1 E2}` to decide whether two reduced lambda expressions `E1` and `E2` equate:

```
{AreEqual lambda(x x) lambda(y y)} % returns true
```

3. Implement the following "let" expression:

```
<expression> ::= let(<atom> <expression> <expression>)
```

meaning let `<atom> = <expression>` in the right hand side `<expression>`, for example,

```
let(x lambda(x x) [x x])
```

would be interpreted as `[lambda(x x) lambda(x x)]` returning the identity combinator.

4. Implement the basic arithmetic operators: plus, minus, times and div, and numbers in the places of expressions, for example,

```
[lambda(x [plus x 2]) 3]
```

would be evaluated to 5, and

```
(let a 5 [lambda(x [plus x a]) 3])
```

would be evaluated to 8.

5. See the professor if you have ideas for other extensions to this assignment and would like extra credit for implementing them.

Due Date:

Received Time	Grade Modification
before Tuesday, 03/10, 6:00PM	+10%
before Wednesday, 03/11, 6:00PM	no modification (on time)
before Thursday 03/12, 6:00PM	-10%
before Saturday, 03/14, 6:00PM	-25%
after Saturday, 03/14, 6:01PM	not accepted

Grading: The assignment will be graded mostly on correctness, but code clarity / readability will also be a factor (comment, comment, comment!). See the professor or TAs, if you have ideas for other extensions for this assignment and would like extra credit for implementing them.

Submission Requirements: Please submit a ZIP file with your code, including a README file. In the README file, place the names of each group member (up to two). Your README file should also have a list of specific features / bugs in your solution. Your ZIP file should be named with your LMS user name(s) as the filename, either userid1.zip or userid1_userid2.zip. Only submit one assignment per pair via LMS.