Implementation of a Functional Programming Language  
Jason Koenig  
Computer Systems Lab 2009-2010

Abstract
Scripting languages have increased greatly in popularity in recent years with the growing power of computers. The trade off of runtime and programmer time is increasing favoring using more runtime. However, most current scripting languages are imperative. A language is developed which is primarily functional in style. The language has novel features which allow the base interpreter to be small in size, will the lack of features such as eval allow the programs to be optimized easily.

Introduction
The purpose of my project is to develop a functional style programming language. The language is similar to Lisp, but contains features to make it friendlier to imperative programmers. The initial version will be interpreted, but I expect to eventually at least partially compile code.

One goal is to make the interpreter as small as possible, allowing the language to easily be embedded in other programs. This will allow my language to be used both on its own, and embedded as a scripting language like Python.

Beyond the implementation, I will also develop a series of tutorials and example programs that will assist in learning my new language. This will be important if my language is to become anything other than a toy language.

Sample Program
let  
  \( \text{X} = \text{5,} \)  
  \( \text{Y} = \text{3+X,} \)  
  \( \text{Z= \{} \text{t| t+2} \} \)  
in  
  \( \text{Z.(X+Y)} \)  # output is 15

Design
In my language, like other functional languages, a program is executed by evaluating the main expression. This expression is usually composed of sub-expressions, which are then composed of sub-expressions, and so on.

Several novel ideas were incorporated, such as using an explicit character for function application (.) rather than whitespace. This simplifies the parser greatly, which was a design goal.

Results
The language currently can perform reduction of complex mathematical expressions. It also has support for a large number of the final operators that are part of the core language. It can perform let reduction, as well as execute simple user defined functions. There is support for imperative programming. Recursive functions work properly. Both a program that reads in source code from a file, as well as one that accepts user input from the command line were created.