Lisp -> Ruby

**PROJECT BY: RUCHIR PATEL. F002BZ1**

Submission date June 4, 2016

**Task:** Convert a "defun lisp" into "def ruby".

**Description:** Run a ruby.rb file, input a valid "defun lisp" function, (lexically binded), and file will output the corresponding ruby function which takes the same amount of parameters and does the same thing as lisp function.

**Language:** ruby

**Partners:** none. Would prefer to work alone on this project.

**Code:** <https://github.com/ruchir594/ruby2lisp>

Note: there are two file dependencies you can run.

1. rb2l.rb  
   This is stripped down version, which will ask you to input a valid lisp function and write the output in ruby\_is.rb  
     
   requires rb2l\_header.rb, condhead.rb, simhead.rb
2. rb2l\_wc.rb  
   This will print a lot of stuff on command prompt which one can understand using the all “filename\_wc.rb” files. These are the files with the same code as their counterpart but with comment lines and print statements.   
     
   requires rb2l\_header\_wc.rb, condhead\_wc.rb, simhead\_wc.rb

**Test Cases:**

Concept: My idea is to read a *defun lisp* function and write something like a language translator. It turned out to be very very challenging and It was more of, observing the pattern, encoding it, running a new test case to realize the code isn’t proper. It was very heuristic at first.

So if you look at the “simhead.rb”, the code is kinda messy, I spend a lot of hours just understanding and carving out a way to solve it. Everything I learned form this exercise is visible in the “condhead.rb”. The code is lean, very well structured compared to the previous code, and also works better.

Approach:

Step 1: extract the name of the function and all the parameters.

Step 2: extract what I call “building blocks”  
  
these building blocks are various bits of string between a pair of bracket ‘(‘ and ‘)’. These will help understand the “scope” and parse the rest of the defun appropriately.

Step 3: squash/simplify the defun.   
  
This is essentially the translation part. I take the syntax of defun and convert it into the appropriate ruby syntax.

For example….  
x is list in lisp, and array in ruby

(car x) -> x[0]

(cdr x) -> x[1..-1]

(null (car x)) -> nil == x[0]

and so on……

I encoded these three steps. Step 3 is yet to be perfected.