$Id: pseudo-code.mm,v 1.1 2018-10-10 15:09:53-07 - - $
PWD: /afs/cats.ucsc.edu/courses/cmps112-wm/Assignments/asg1-scheme-sbi
URL: http://www2.ucsc.edu/courses/cmps112-wm/:/Assignments/asg1-scheme-sbi/

The data structure consists of a recursively nested list :

- The top level list consists of a sequence of lines. Each line is pointed at by the **car** of a cell in the opt level list.
- Each line consists of a line number, an optional label, which is always a **symbol?**, and an optional statement, which is always a **pair?**. Use **null?** to determine whether not something exists. Do not use **list?**.
- A statement consists of a keyword followed by operands, mostly expressions.
- An expression uses prefix notation in standard Scheme format.

A suggested outline and description of some of the functions follows :

(a) After reading in the program, make one pass over the top level, checking for a label in each line. Each label should be inserted into the label hash with a pointer to the top level node (not the line).

(b) Write a function **interpret-program** takes the top level list as an argument and checks to see if there is a statement.
- If there is no statement, call it recursively with the **cdr** of the top level node.
- If there is a statement, look up the keyword in the statement hash and call **interpret-***statement* , where *statement* is the keyword found in the statement.
- This funcion should return null for a statement that is not a control transfer, or for a statement that is a control transfer that is not taken.
- If this function returns a null then call **interpret-program** recursively with the **cdr**, as explained above.
- If this function is a successful control transfer, it should return the label to which to transfer, and then **interpret-program** calls itself recursively with the associated line.

(c) Write separate functions **interpret-***statement* for each one of the keyword in the language.

(d) The function **evaluate-expression** is called by a statement interpreter.
- It looks up the function in the function table.
- It uses **map** to call **evaluate-expression** for each of the arguments to the function.
- Then use **apply** to apply the function to the list of results obtained.
- Subscripting arrays will require a special case.

(e) Implementation may use one or two symbol tables : Functions and identifiers may be placed into the same table, or they may be separated into two tables. There is an ambiguity in the notation for calling a function and subscripting an array.