Homework, Week 1, part B

lightprogramming.org

September 9, 2016

The software development process has four steps. This week's homework will walk you through all four steps. The steps are: (1) analysis, (2) design, (3) implementation, and (4) testing. The process repeats all four steps, one after another, many times before the software is complete.

The purpose of implementation is to write the software using the design. I have written a simple addition test program below. Copy this into a file named "add-test.lisp", load it into your clisp environment, and run it.

You do not have to know what this does. This short code has seven variables. You should understand what each variable does. They are: number-of-questions, number-correct, question-counter, a, b, c, and d. Do an internet search for "common lisp function defparameter" and "common lisp function let."

```lisp
;;; add-test.lisp

(print "This is add-lisp. Evaluate (start-test) to start the test.")

(defun start-test ()
  (defparameter number-of-questions 10)
  (defparameter number-correct 0)
  (defparameter question-counter 1)
  (format t "Starting the addition test, you have ~a questions.~%"
    number-of-questions)
  (run-test))

(defun addition-problem ()
  (let* ((a (random 11))
         (b (random 11))
         (c (+ a b))
         (d (read (format t "What is ~a + ~a? " a b))))
    (cond ((= c d)
           (format t "Correct~%")
            1)
          (t (format t "The answer is ~a~%" c)
              0))))

(defun run-test ()
    (cond
      ((zerop number-of-questions)
       (format t "You got ~a correct and made a ~a.~%" number-correct (* 100
    (/ number-correct 10.0)))
       T)
      (t (format t "Question ~a. " question-counter)
         (decf number-of-questions)
         (incf number-correct (addition-problem))
                 (incf question-counter)
         (run-test))))
```

Testing Load this file into your lisp with `(load "add-test.lisp")` and evaluate `(start-test)`. Does it do what it is supposed to do? Does it meet all the requirements you identified? Does it do anything that you did not identify as a requirement?

Answers Ordinarily, you should always give your variables and functions understandable, English-like names. I have done this with the main parts: start-test, run-test, and addition-problem. What these do should be completely evident from their names. I have given three of the variables the same kind of name: number-of-questions, number-correct, and question-counter. Again, the values these contain should be completely evident. I have violated this rule with the variables named a, b, c, and d. The reason is that they are set and used in a compact piece of code so that, again, what they do should be completely evident. Variables a and b are the two numbers to be added; c is the sum of a and b; and d is the answer supplied by the student. By now, you have already seen `random` and `+`. The `read` function reads a number from the keyboard, and "returns" it to the calling function, exactly like `random` and `+`. These two also "return" a value to the calling function. The "calling function" is `let`.