Program by Design - DESIGN RECIPE - VERSION 1

It is good practice to follow the Design Recipe for all functions that you write.

Step 1 - problem analysis and data definition

- Consider your problem; consider the kinds of data involved in your problem. Determine if you need to define any new kinds of data, and develop data definitions to do so as applicable.

Step 2 - signature/purpose/header

- First develop a signature comment, including a nicely-descriptive name for your function, the **types** of expressions it expects, and the **type** of expression is produces. For example,

```
; signature: rect-area: number number -> number
```

- Then develop a purpose comment, **describing** what the function expects and **describing** what it produces. For example,

```
; purpose: expects the length and width of a rectangle,
;          and produces the area of that rectangle
```

- Now write the function header, giving a good, descriptive name for each parameter variable. Use `...` as a stub for the function body at this point.

Step 3 - develop specific examples/tests

- Now develop `check-expect` (or `check-within`, or other `check-` operation) expressions expressing specific examples of your function that you devise **before** writing your function body.
  - (These may be **placed** before or after your actual function definition, but you should **create** these **before** writing the function body.)
  - For example,

```
(check-expect (rect-area 3 4)
              12)
```

- How many check-expect expressions should you have? That is an excellent question, and does depend on the situation.
  - The **basic rule of thumb** is that you need a specific example/`check-` expression for each "case" or category of data that may occur, and for each "boundary" between categories... and you can always add more if you'd like!

Step 4 - decide which body template is appropriate

- Replace the `...` that is currently your function body with an appropriate template, based on the problem type.

Step 5 - Develop/complete the function's body

- Either replace the `...` that is currently your function body, or finish filling in/completing the body template you developed in Step 4.

Step 6 - Run the tests

- Click the Run button! 8-)
- Note that you may include as many additional calls or tests of your function as you would like after its definition.