Design Recipe - Common Errors

Purpose:

- Remember to reference parameters in your purpose statement.
- Remember to have all parameters in your function header as well.

Contract:

- Ensure data types are as specific as possible. For example, using "Int" over "Nat", where it is appropriate.
- Requirements section belongs under the contract, not the purpose
- Remember that lists have brackets around them in contracts (e.g. "(listof Num)")

Question 2 (listfun)

eval-poly:

- Contract must specify that the list is non-empty

Question 3 (rainbow)

unicorn and leprechaun:

- For their contracts, using the data types, Colour and Rainbow is the recommended method. However, if you did not use them, a requirements section is necessary. See the example below:

```
1 ;; unicorn: Sym (listof Sym) -> (listof Sym)
2 ;; requires: * colour is one of 'red, 'orange, 'yellow, 'green, 'blue,
3 ;;             'indigo, 'violet
4 ;;           * elements in rainbow are 'red, 'orange, 'yellow, 'green,
5 ;;             'blue, 'indigo, or 'violet
6 ;;           * elements in rainbow are unique and in the order:
7 ;;             'red, 'orange, 'yellow, 'green, 'blue, 'indigo, 'violet
```

Question 4 (titles)

capitalize-after:

- Should have used (alt-listof Char) as input in the contract, not (listof Char)

title-case:

- title-case should have been used as a wrapper function around capitalize-after

Question 5 (div-by-3)

nat3-template:

- Nat3 is the correct input for the contract, NOT Nat
- In the recursive step, subtraction should be by 3
- Please review the solution to make sure you got all base cases if you lost template marks

div-by-3?:

- Parameter name like Nat3, Nat, etc. are bad style. Please review the naming section in the style guide for more information.
- Using "(< n 0)" as a question for a conditional violates the contract condition that the input, n, is a Nat3