The Design Recipe

For the word problems below, assume you have `animalA` and `animalB` defined in your code.

Directions: Define a function called `is-old`, which consumes a Row of the animals table and *computes* whether it is more than 12 years old.

Contract and Purpose Statement

Every contract has three parts…

```
# is-old:: (r :: Row) -> Boolean
```
function name | *domain* | *range*

```
# Consumes an animal, and computes whether it's age is > 12
```
what does the function do?

Examples

Write some examples, then circle and label what changes…

examples:

```
is-old ( "animalA" ) is animalA["age"] > 12
```
function name | *input(s)* | *what the function produces*

```
is-old ( "animalB" ) is animalB["age"] > 12
```
function name | *input(s)* | *what the function produces*

end

Definition

Write the definition, giving variable names to all your input values…

```
fun is-old( r ):
```
function name | *variable(s)*

```
  r["age"] > 12
```
what the function does with those variable(s)

end

Directions: Define a function called `name-has-s`, which returns true if an animal's name contains the letter "s"

Contract and Purpose Statement

Every contract has three parts…

```
# name-has-s:: (r :: Row) -> Boolean
```
function name | *domain* | *range*

```
# Consumes an animal, and computes whether its name contains an "s"
```
what does the function do?

Examples

Write some examples, then circle and label what changes…

examples:

```
name-has-s ( "animalA" ) is string-contains(animalA["name"], "s")
```
function name | *input(s)* | *what the function produces*

```
name-has-s ( "animalB" ) is string-contains(animalB["name"], "s")
```
function name | *input(s)* | *what the function produces*

end

Definition

Write the definition, giving variable names to all your input values…

```
fun name-has-s( r ):
```
function name | *variable(s)*

```
  string-contains(r["name"], "s")
```
what the function does with those variable(s)

end