Design Recipe Exercise Answer Key

target-leap

1. The first example doesn't work. It would work better as

   `target-leap(100) is 100 * 2` or `target-leap(100) is 2 * 100`

2. The second example is incorrect according to the problem statement. It could be

   `target-leap(40) is 40 * 2` or `target-leap(40) is 2 * 40`

3. In the definition, the name of the function should be `target-leap`.

4. The variable name in the definition should be consistent (either `x` or `x-coor`).

5. The body of the function definition should be

`x-coor * 2`.

is-offscreen

1. The Purpose Statement should include "return true if the coordinate is less than -50 or greater than 690".

2. Both examples should show the work, not just the answer:

   a. `is-offscreen(60) is (60 < -50) or (60 > 690)`

   b. `is-offscreen(800) is (800 < -50) or (800 > 690)`

3. In the definition, the function name should be `is-offscreen` instead of `is-off-screen`.

4. The function definition should use the function `or` instead of `and`.

calc-pencils

1. Both examples should multiply by 5.

2. The variable should be more descriptive: s, or students, to represent the number of students.

3. In the definition, the function name should be `calc-pencils`.

circle-area

1. `pi` is not a built-in value, so it should be replaced with an approximation such as `3.14` or `(22 / 7)`.

2. If using `(22 / 7)` for `pi`, the function body could be

   `num-sqr(diameter / 2) * (22 / 7))`

check-total

1. The examples should use the function name `check-total` instead of `total`.

2. Both examples have an extra input that isn't attached to an operator or function:

   `((0.2 * 20) + 20)`

3. The `*` operator must be used instead of `x` to show multiplication in the examples:

   `((0.20 * 56.67) + 56.67)`

4. The function body should have the `*` and `+` operators reversed:

   `(0.20 * food-total) + food-total`

have-enough-carpet

1. The range of the function should be a Boolean.

2. The example inputs should not be in an extra set of parentheses.

3. Both the examples and the function definition should use `<=` instead of `<`.

4. The example inputs should be separated by a comma.

have-enough-cash

1. The domain of the function should be a Number (representing the price), not a String.
2. The two examples should give numbers as an input and test if they are less than 1.50. For instance,

   ```
   have-enough-cash(2.50) is 2.50 <= 1.50
   ```

3. The variable name in the function body can be `item` , but a more accurate name would be `price` or `cost` .

equal-length

1. The function body should be:

   ```
   string-length(string1) == string-length(string2)
   ```

2. The inputs in the examples should be in quotes— `"yes"` , `"no"` .

flower-name

1. The purpose statement should read "Takes in the color and returns the name of that flower".
2. The second example should be

   ```
   flower-name("purple") is "tulip"
   ```

3. In the examples, all the colors and flower names should be Strings, written inside quotation marks.

is-long-name

1. Both examples should use the function `string-length` , not `string-equal` .
2. The examples should check if the name is longer than 20 characters, not 10.
3. The function name in the definition should be `is-long-name` .
4. The body of the function should be

   ```
   string-length(name) < 20
   ```

scale-image

1. The purpose statement doesn't specify which strings matter, or how much to scale by.

   The examples do not use the `scale` function at all, and instead change the parameters of the image. The first example should be:

   ```
     scale-image(circle(5, "solid", "red"), "bigger") is
             scale(2, circle(5, "solid", "red"))
   ```

2. The function name in the second example is incorrect.

state-tax

1. The domain for the function should be `String, Number` to account for both the state and the price of the item.
2. The function name in both examples should be `state-tax` .
3. The example inputs (`"Delaware"` and `"Georgia"`) should be Strings.
4. Examples should include a numerical price instead of the variable name `price` .
5. The examples should use `*` not `+` .
6. The function variable name should not contain spaces and must be consistent throughout the function definition. It should instead be `price` .

late-to-class

1. Both examples should include 4 numbers as inputs.
2. In the first example, `<` should be used in place of `>` .
3. Both examples and the function definition should calculate distance based on the 4 inputs, such as:

```
late-to-class(40, 55, 80, 100) is 25 < distance(40, 55, 80, 100)
```

4. The two examples should be different from each other. Since the function returns a Boolean, good practice would be to make one example that is true and another that is false.