Name: ____

BOOTSTRAP REACTIVE

Student Workbook

Spring, 2021 - Pyret Edition

BOOTSTRAP

Equity • Scale • Rigor

Workbook v1.5

Brought to you by the Bootstrap team:

- Emmanuel Schanzer
- Emma Youndtsmith
- Kathi Fisler
- Shriram Krishnamurthi
- Dorai Sitaram
- Joe Politz
- Ben Lerner

Visual Designer: Colleen Murphy

Introduction to Programming

The **Editor** is a software program we use to write Code. Our Editor allows us to experiment with Code on the right-hand side, in the **Interactions Area**. For Code that we want to *keep*, we can put it on the left-hand side in the **Definitions Area**. Clicking the "Run" button causes the computer to re-read everything in the Definitions Area and erase anything that was typed into the Interactions Area.

Data Types

Programming languages involve different ***data types***, such as Numbers, Strings, Booleans, and even Images.

- Numbers are values like `1`, `0.4`, `1/3`, and `-8261.003`.
    - Numbers are *usually* used for quantitative data and other values are *usually* used as categorical data.
    - In Pyret, any decimal *must* start with a 0. For example, `0.22` is valid, but `.22` is not.
- Strings are values like `"Emma"`, `"Rosanna"`, `"Jen and Ed"`, or even `"08/28/1980"`.
    - All strings *must* be surrounded in quotation marks.
- Booleans are either `true` or `false`.

All values evaluate to themselves. The program `42` will evaluate to `42`, the String `"Hello"` will evaluate to `"Hello"`, and the Boolean `false` will evaluate to `false`.

Operators

Operators (like `+`, `-`, `*`, `<`, etc.) work the same way in Pyret that they do in math.

- Operators are written between values, for example: `4 + 2`.
- In Pyret, operators must always have a space around them. `4 + 2` is valid, but `4+2` is not.
- If an expression has different operators, parentheses must be used to show order of operations. `4 + 2 + 6` and `4 + (2 * 6)` are valid, but `4 + 2 * 6` is not.

Applying Functions

Applying functions works much the way it does in math. Every function has a name, takes some inputs, and produces some output. The function name is written first, followed by a list of ***arguments*** in parentheses.

- In math this could look like $f(5)$ or $g(10, 4)$.
- In Pyret, these examples would be written as `f(5)` and `g(10, 4)`.
- Applying a function to make images would look like `star(50, "solid", "red")`.
- There are many other functions, for example `num-sqr`, `num-sqrt`, `triangle`, `square`, `string-repeat`, etc.

Functions have ***contracts***, which help explain how a function should be used. Every contract has three parts:

- The *Name* of the function - literally, what it's called.
- The *Domain* of the function - what *types of values* the function consumes, and in what order.
- The *Range* of the function - what *type of value* the function produces.

Numbers and Strings

Make sure you've loaded the code.pyret.org, (CPO) editor, clicked "Run", and are working in the ***Interactions Area***.

Numbers

1) Try typing `42` into the Interactions Area and hitting "Enter". What is the largest number the editor can handle?

____

2) Try typing `0.5` . Then try typing `.5` . Then try clicking on the answer. Experiment with other decimals. Explain what you understand about how decimals work in this programming language.

____

____

3) What happens if you try a fraction like `1/3` ?

____

4) Try writing negative integers, fractions and decimals.

Strings

String values are always in quotes.

5) Is `42` the same as `"42"` ? Why or why not? Write your answer below:

____

6) Try typing your name *(in quotes!)* .

7) Try typing a sentence like "I'm excited to learn to code!" *(in quotes!)* .

8) Try typing your name with the opening quote, but *without the closing quote.* Read the error message!

9) Now try typing your name *without any quotes.* Read the error message!

10) Explain what you understand about how strings work in this programming language.

____

Operators

11) Just like math, Pyret has ***operators*** like `+` , `-` , `*` and `/` . Try typing in `4 + 2` , and then `4+2` (without the spaces). What can you conclude from this?

____

12) Type in the following expressions, one at a time: `4 + 2 + 6` , `4 + 2 * 6` , `4 + (2 * 6)` . What do you notice?

____

13) Try typing in `4 + "cat"` , and then `"dog" + "cat"` . What can you conclude from this?

____

Booleans

Boolean-producing expressions are yes-or-no questions and will always evaluate to either `true` ("yes") or `false` ("no"). What will each of the expressions below evaluate to? *Write down your prediction in the blanks provided and then type the code into the interactions area to see what it returns.*

	Prediction:	Computer Returns:		Prediction:	Computer Returns:
1)`3 <= 4`			2)`"a" > "b"`		
3)`3 == 2`			4)`"a" < "b"`		
5)`2 < 4`			6)`"a" == "b"`		
7)`5 >= 5`			8)`"a" <> "a"`		
9)`4 >= 6`			10)`"a" >= "a"`		
11)`3 <> 3`			12)`"a" <> "b"`		

13) In your own words, describe what `<` does.

____

14) In your own words, describe what `>=` does.

____

15) In your own words, describe what `<>` does.

____

	Prediction:	Computer Returns:
16)`string-contains("catnap", "cat")`		
17)`string-contains("cat", "catnap")`		

18) How many **Numbers** are there in the entire universe? ____________________

19) How many **Strings** are there in the entire universe? ____________________

20) How many **Images** are there in the entire universe? ____________________

21) How many **Booleans** are there in the entire universe? ____________________

Applying Functions

Type this line of code into the interactions area and hit "Enter":

```
triangle(50, "solid", "red")
```

1	What is the name of this function?	
2	What did the expression evaluate to?	
3	How many arguments does `triangle` expect?	
4	What data type does the `triangle` function produce? (Numbers? Strings? Booleans?)	

Catching Bugs

The following lines of code are all BUGGY! Read the code and the error messages to identify the mistake.

5) `triangle(20, "solid" "red")`

Pyret didn't understand your program around

triangle(20, "solid" **"red"**)

Can you spot the mistake? ______________________________

6) `triangle(20, "solid")`

This <u>application expression</u> errored:

triangle (*20* , *"solid"*)

<u>*2 arguments*</u> were passed to the <u>**operator**</u> . The <u>**operator**</u> evaluated to a function accepting 3 parameters. An <u>application expression</u> expects the number of parameters and <u>*arguments*</u> to be the same.

Can you spot the mistake? ______________________________

7) `triangle(20, 10, "solid", "red")`

This <u>application expression</u> errored:

triangle (20, 10, "solid", *"red"*)`

<u>*4 arguments*</u> were passed to the <u>**operator**</u> . The <u>**operator**</u> evaluated to a function accepting 3 parameters. An <u>application expression</u> expects the number of parameters and <u>*arguments*</u> to be the same.

Can you spot the mistake? ______________________________

8) `triangle (20, "solid", "red")`

Pyret thinks this code is probably a function call:

triangle *(20, "solid", "red")*

Function calls must not have space between the <u>**function expression**</u> and the <u>*arguments*</u> .

Can you spot the mistake? ______________________________

Domain and Range

My Definition

Facts & Characteristics

Domain

Examples

Non-examples

My Definition

Facts & Characteristics

Range

Examples

Non-examples

Practicing Contracts: Domain & Range

Consider the following contract:

```
is-beach-weather :: Number, String -> Boolean
```

1) What is the **Name** of this function? ______________________________

2) How many arguments are in this function's **Domain**? ______________________________

3) What is the **type** of this function's **first argument**? ______________________________

4) What is the **type** of this function's **second argument**? ______________________________

5) What is the **Range** of this function? ______________________________

6) Circle the expression below that shows the correct application of this function, based on its contract.

A. `is-beach-weather(70, 90)`

B. `is-beach-weather(80, 100, "cloudy")`

C. `is-beach-weather("sunny", 90)`

D. `is-beach-weather(90, "stormy weather")`

Consider the following contract:

```
cylinder :: Number, Number, String -> Image
```

7) What is the **Name** of this function? ______________________________

8) How may arguments are in this function's **Domain**? ______________________________

9) What is the **type** of this function's **first argument**? ______________________________

10) What is the **type** of this function's **second argument**? ______________________________

11) What is the **type** of this function's **third argument**? ______________________________

12) What is the **Range** of this function? ______________________________

13) Circle the expression below that shows the correct application of this function, based on its contract.

A. `cylinder("red", 10, 60)`

B. `cylinder(30, "green")`

C. `cylinder(10, 25, "blue")`

D. `cylinder(14, "orange", 25)`

Matching Expressions and Contracts

Match the contract (left) with the expression described by the function being used (right).

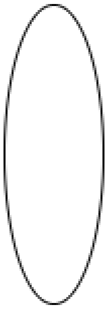
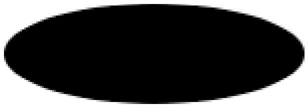
Contract			Expression
`# make-id :: String, Number -> Image`	1	A	`make-id("Savannah", "Lopez", 32)`
`# make-id :: String, Number, String -> Image`	2	B	`make-id("Pilar", 17)`
`# make-id :: String -> Image`	3	C	`make-id("Akemi", 39, "red")`
`# make-id :: String, String -> Image`	4	D	`make-id("Raïssa", "McCracken")`
`# make-id :: String, String, Number -> Image`	5	E	`make-id("von Einsiedel")`

Contract			Expression
`# is-capital :: String, String -> Boolean`	6	A	`show-pop("Juneau", "AK", 31848)`
`# is-capital :: String, String, String -> Boolean`	7	B	`show-pop("San Juan", 395426)`
`# show-pop :: String, Number -> Image`	8	C	`is-capital("Accra", "Ghana")`
`# show-pop :: String, String, Number -> Image`	9	D	`show-pop(3751351, "Oklahoma")`
`# show-pop :: Number, String -> Number`	10	E	`is-capital("Albany", "NY", "USA")`

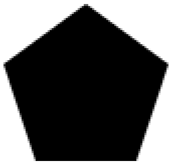
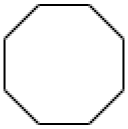
Using Contracts

Use the contracts to write expressions to generate images similar to those pictured.

`ellipse :: Number, Number, String, String -> Image`

	
	
What changes with the first number?	
What about the shape changes with the second Number?	
Write an expression using `ellipse` to produce a circle.	

`regular-polygon :: Number, Number, String, String -> Image`

	
	
What changes with the first Number?	
What about the shape changes with the second Number?	
Use `regular-polygon` to write an expression for a square!	
How would you describe a **regular polygon** to a friend?	

Triangle Contracts

1) What kind of triangle does the `triangle` function produce? ____________________

There are lots of other kinds of triangles! And Pyret has lots of other functions that make triangles!

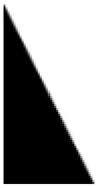
```
triangle :: (size:: Number, style :: String, color :: String) -> Image
right-triangle :: (base::Number, height::Number, style::String, color::String) -> Image
isosceles-triangle :: (leg::Number, angle::Number, style::String, color::String) -> Image
```

2) Why do you think `triangle` only needs one number, while `right-triangle` and `isosceles-triangle` need two numbers and `triangle-sas` needs three?

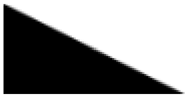
____

____

3) Write `right-triangle` expressions for the images below. *One argument for each should be* `100` .



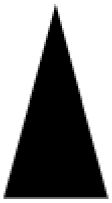
____



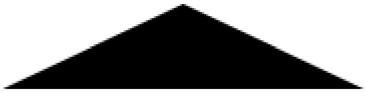
____

4) What do you think the numbers in `right-triangle` represent? ____________________

5) Write `isosceles-triangle` expressions for the images below. *1 argument for each should be* `100` .



____

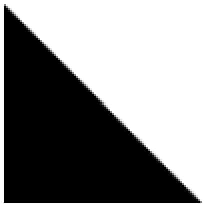


____

6) What do you think the numbers in `isosceles-triangle` represent?

____

7) Write 2 expressions that would build **right-isosceles** triangles. Use `right-triangle` for one expression and `isosceles-triangle` for the other expression.



____

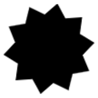
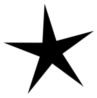
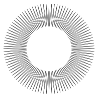
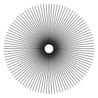
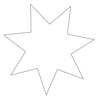
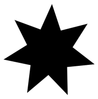
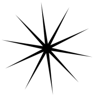
____

Radial Star

```
radial-star :: (

      points :: Number,
      inner-radius :: Number,
      full-radius :: Number,
      style :: String,
      color :: String
  ) -> Image
```

Using the detailed contract above, match each image to the expression that describes it.

Image			Expression
	1	A	`radial-star(5, 50, 200, "solid", "black")`
	2	B	`radial-star(7, 100, 200, "solid", "black")`
	3	C	`radial-star(7, 100, 200, "outline", "black")`
	4	D	`radial-star(10, 150, 200, "solid", "black")`
	5	E	`radial-star(10, 20, 200, "solid", "black")`
	6	F	`radial-star(100, 20, 200, "solid", "black")`
	7	G	`radial-star(100, 100, 200, "outline", "black")`

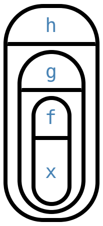
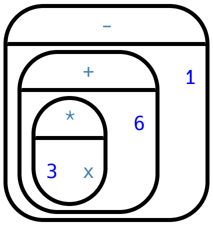
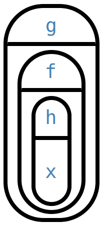
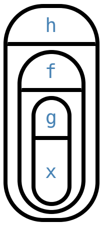
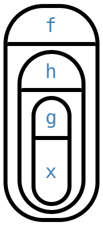
What’s on your mind?

Diagramming Function Composition

`f :: Number -> Number`
`Consumes a number, multiplies by 3 to produce the result`
	`g :: Number -> Number`
	`Consumes a number, adds six to produce the result`
		`h :: Number -> Number`
		`Consumes a number, subtracts one to produce the result`

$f(x) = 3x$	$g(x) = x + 6$	$h(x) = x - 1$

For each function composition diagrammed below, translate it into the equivalent Circle of Evaluation for Order of Operations. Then write expressions for *both* versions of the Circles of Evaluation, and evaluate them for $x = 4$. The first one has been completed for you.

Function Composition	Order of Operations	Translate & Evaluate	
1) 		Composition:	`h(g(f(x)))`
		Operations:	`((3 * x) + 6) - 1`
		Evaluate for x = 4	$h(g(f(4))) = 17$
2) 		Composition:	
		Operations:	
		Evaluate for x = 4	____________________
3) 		Composition:	
		Operations:	
		Evaluate for x = 4	____________________
4) 		Composition:	
		Operations:	
		Evaluate for x = 4	______________________

Function Composition — Green Star

1) Draw a Circle of Evaluation and write the Code for a **solid, green star, size 50**.

Circle of Evaluation:

Code: ____

Using the star described above as the **original**, draw the Circles of Evaluation and write the Code for each exercise below.

2) A solid, green star, that is triple the size of the original (using `scale`) **Circle of Evaluation:** **Code:** ______________________	3) A solid, green star, that is half the size of the original (using `scale`) **Circle of Evaluation:** **Code:** ______________________
4) A solid, green star of size 50 that has been rotated 45 degrees counter-clockwise **Circle of Evaluation:** **Code:** ______________________	5) A solid, green star that is 3 times the size of the original **and** has been rotated 45 degrees **Circle of Evaluation:** **Code:** ______________________

Function Composition — Your Name

You’ll be investigating these functions with your partner:

```
# text :: String, Number, String -> Image              # frame :: Image -> Image
# flip-horizontal :: Image -> Image                    # above :: Image, Image -> Image
# flip-vertical :: Image -> Image                      # beside :: Image, Image -> Image
```

1) In the editor, write the code to make an image of your name in big letters in a color of your choosing using `text` . Then draw the Circle of Evaluation and write the Code that will create the image.

Circle of Evaluation:

Code: ____

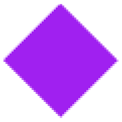
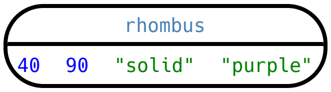
Using the "image of your name" described above as the **original** , draw the Circles of Evaluation and write the Code for each exercise below. Test your ideas in the editor to make sure they work.

2) The framed "image of your name". **Circle of Evaluation:** **Code:** ________________________________	3) The "image of your name" flipped vertically. **Circle of Evaluation:** **Code:** ________________________________
4) The "image of your name" above "the image of your name" flipped vertically. **Circle of Evaluation:** **Code:** ________________________________	5) The "image of your name" flipped horizontally beside "the image of your name". **Circle of Evaluation:** **Code:** ________________________________

Function Composition — scale-xy

You'll be investigating these two functions with your partner:

```
# scale-xy :: Number, Number, Image -> Image                # overlay :: Image, Images -> Image
```

The Image:	Circle of Evaluation:	Code:
	 rhombus 40 90 "solid" "purple"	`rhombus(40, 90, "solid", "purple")`

Starting with the image described above, write the Circles of Evaluation and Code for each exercise below. Be sure to test your code in the editor!

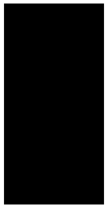
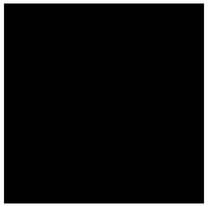
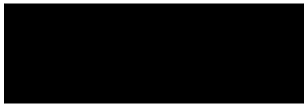
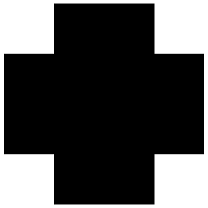
1) A purple rhombus that is stretched 4 times as wide. **Circle of Evaluation:** **Code:** ______________________________	2) A purple rhombus that is stretched 4 times as tall **Circle of Evaluation:** **Code:** ______________________________
3) The tall rhombus overlayed on the wide rhombus. **Circle of Evaluation:** **Code:** ______________________________	★: Overlay a red rhombus onto the last image you made. **Circle of Evaluation:** **Code:** ______________________________

More than one way to Compose an Image!

Read through these 4 expressions and try to picture the images they are composing. If you're not sure what they'll look like, type them into the interactions area of your editor and see if you can figure out how the code connects to the image.

```
beside(rectangle(200, 100, "solid", "black"), square(100, "solid", "black"))
scale-xy(1, 2, square(100, "solid", "black"))
scale(2, rectangle(100, 100, "solid", "black"))
above(
    rectangle(100, 50, "solid", "black"),
    above(
        rectangle(200, 100,  "solid", "black"),
        rectangle(100, 50, "solid", "black")))
```

For each image below, identify 2 expressions that could be used to compose it. *The bank of expressions at the top of the page includes one possible option for each image.*

1		• `rotate(90, rectangle(200, 100, "solid", "black"))` • ______________________ • ______________________
2		• `above(rectangle(200, 100, "solid", "black"), rectangle(200, 100, "solid", "black"))` • ______________________ • ______________________
3		• `scale(0.5, rectangle(600, 200, "solid", "black"))` • ______________________ • ______________________
★		• `overlay(rectangle(100, 200, "solid", "black"), rectangle(200, 100, "solid", "black"))` • ______________________ • ______________________

Defining Values

In math, we use **values** like -98.1, $2/3$ amd 42. In math, we also use **expressions** like 1×3, $\sqrt{16}$, and $5 - 2$. These evaluate to results, and typing any of them in as code produces some answer.

Math also has **definitions**. These are different from values and expressions, because they *they do not produce results*. Instead, they simply create names for values, so that those names can be re-used to make the Math simpler and more efficient.

Definitions always have both a name and an expression. The name goes on the left and the value-producing expression goes on the right, separated by an equals sign:

$$x = 4$$
$$y = 9 + x$$

The name is defined to be the result of evaluating the expression. Using the above examples, we get "x is defined to be 4, and y is defined to be 13". **Important: there is no "answer" to a definition**, and typing in a definition as code will produce no result.

Notice that *definitions can refer to previous definitions*. In the example above, the definition of `y` refers to `x`. But `x`, on the other hand, *cannot* refer to `y`. Once a value has been defined, it can be used in later expressions.

In Pyret, these definitions are written the *exact same way*:
Try typing these definitions into the Definitions Area on the left, clicking "Run", and then *using* them in the Interactions Area on the right.

```
x = 4
```

```
y = 9 + x
```

Just like in math, definitions in our programming language can only refer to previously-defined values.

Here are a few more value definitions. Feel free to type them in, and make sure you understand them.

```
x = 5 + 1
```

```
y = x * 7
```

```
food = "Pizza!"
```

```
dot = circle(y, "solid", "red")
```

Defining Values - Explore

Open the Defining Values Starter File and click run.

1) What do you notice?

____

____

____

2) What do you wonder?

____

____

____

Look at the expressions listed below. Think about what you expect each of them to produce. Then, test them out one at a time in the Interactions Area.

- `x`
- `x + 5`
- `y - 9`
- `x * y`
- `z`
- `t`
- `gold-star`
- `my-name`
- `swamp`
- `c`

3) What have you learned about defining values?

____

____

____

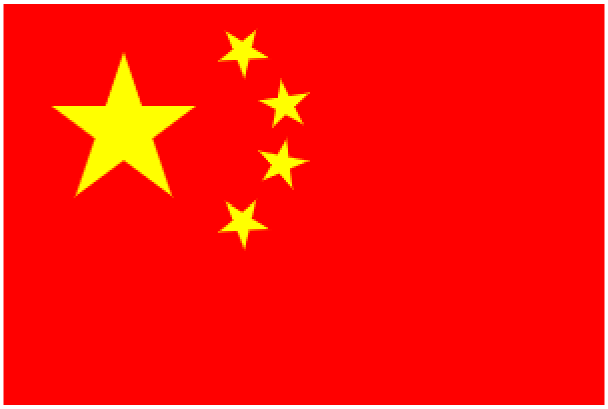
____

4) Define at least 2 more variables in the definitions area, click run and test them out. Once you know they're working, record the code you used below.

____

____

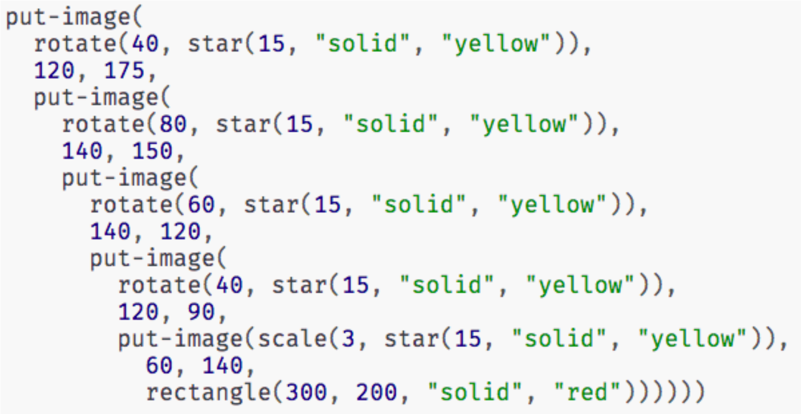
Defining Values - Chinese Flag



1) What image do you see repeated in the flag? ______________________________

2) Highlight or circle all instances of the structure that makes the repeated image in the code below.

3) In the code below, highlight or circle all instances of the expression for that image.



```
put-image(
  rotate(40, star(15, "solid", "yellow")),
  120, 175,
  put-image(
    rotate(80, star(15, "solid", "yellow")),
    140, 150,
    put-image(
      rotate(60, star(15, "solid", "yellow")),
      140, 120,
      put-image(
        rotate(40, star(15, "solid", "yellow")),
        120, 90,
        put-image(scale(3, star(15, "solid", "yellow")),
          60, 140,
          rectangle(300, 200, "solid", "red"))))))
```

4) Write the code to define a value for the repeated expression.

____

5) Open the Chinese flag starter file (Pyret) and click Run.

Then type `china` into the interactions area and click **Enter** .

6) **Save a copy** of the file, and simplify the flag code using the value you defined. Click Run, and confirm that you still get the same image as the original.

7) Now change the color of all of the stars to black, in both files. Then change the size of the stars.

8) Why is it helpful to define values for repeated images?

____

____

Challenge:

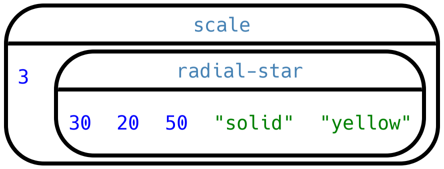
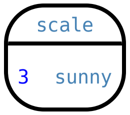
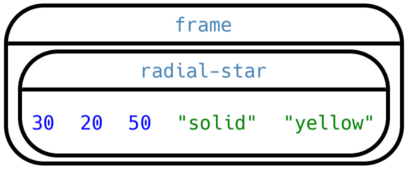
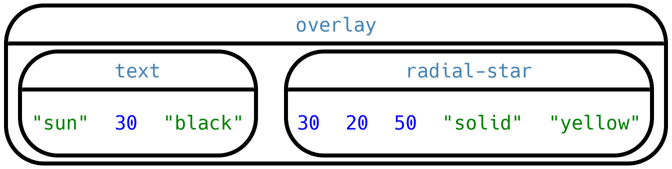
- This file uses a function we haven't seen before! What is it? ______________________
- Can you figure out its contract? *Hint: Focus on the last instance of the function.*

____

Why Define Values?

1) Complete the table using the first row as an example.

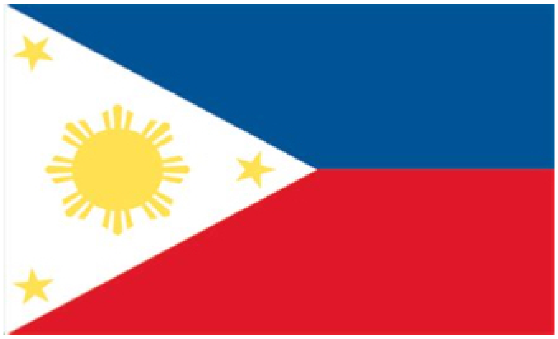
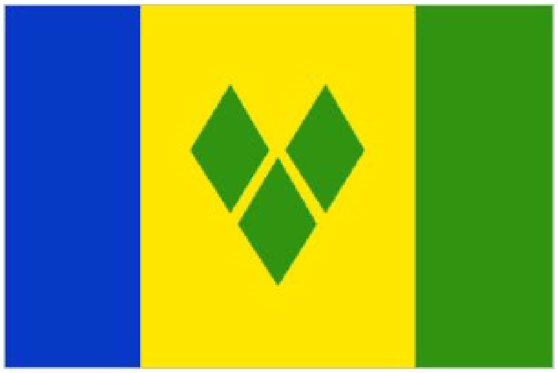
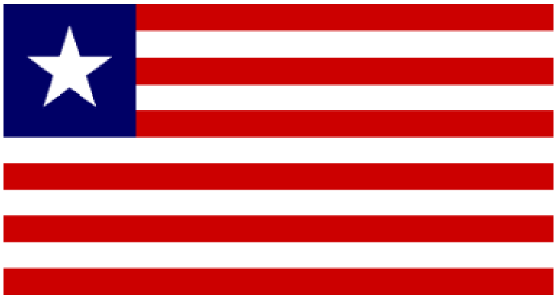
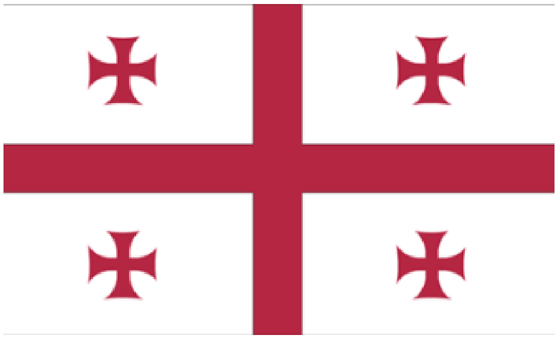
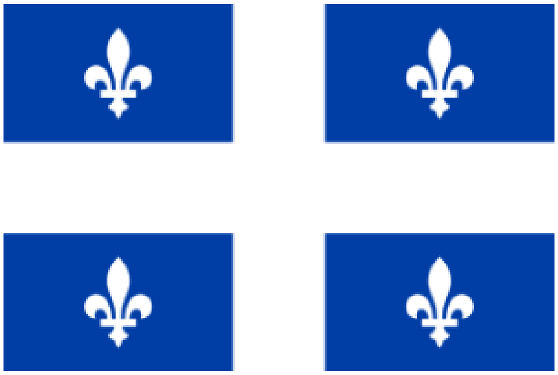
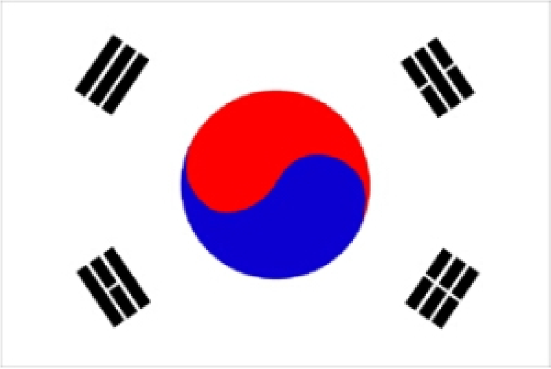
2) Write the code to define the value of `sunny`. ______________________________

Original Circle of Evaluation & Code	→	Use the *defined value* `sunny` to simplify!
	→	
Code: `scale(3, radial-star(30, 20, 50, "solid", "yellow"))`	→	Code: `scale(3, sunny)`
	→	
Code: `frame(radial-star(30, 20, 50, "solid", "yellow"))`	→	Code:
 overlay text "sun" 30 "black" radial-star 30 20 50 "solid" "yellow"	→	
Code: `overlay(text("sun", 30, "black"), radial-star(30, 20, 50, "solid", "yellow"))`	→	Code:

3) Test your code in the editor and make sure it produces what you would expect it to.

Which Value(s) Would it Make Sense to Define?

For each of the images below, identify which element(s) you would want to define before writing code to compose the image. Hint: what gets repeated?

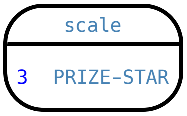
Philippines	St. Vincent & the Grenadines
	
______________________	______________________
Liberia	Republic of Georgia
	
______________________	______________________
Quebec	South Korea
	
______________________	______________________

Writing Code using Defined Values

1) On the line below, **write the Code** to define `PRIZE-STAR` as a pink, outline star of size 65.

____

Using the `PRIZE-STAR` definition from above, draw the Circle of Evaluation and write the Code for each of the exercises. One Circle of Evaluation has been done for you.

2 The outline of a pink star that is three times the size of the original (using `scale`) **Circle of Evaluation:**  	3 The outline of a pink star that is half the size of the original (using `scale`) **Circle of Evaluation:**
Code:	**Code:**
4 The outline of a pink star that is rotated 45 degrees *(It should be the same size as the original.)* **Circle of Evaluation:**	5 The outline of a pink star that is three times as big as the original **and** has been rotated 45 degrees **Circle of Evaluation:**
Code:	**Code:**

6) How does defining values help you as a programmer?

____

____

____

Estimating Coordinates

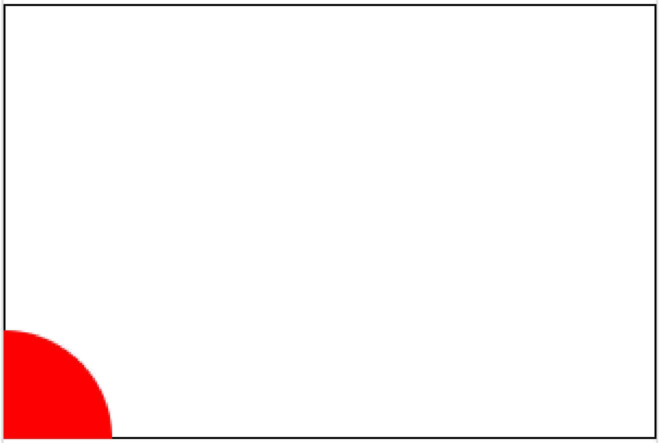
Think of the background image as a sheet of graph paper with the origin (0,0) in the bottom left corner.
The numbers in `put-image` specify a point on that graph paper, where the center of the top image should be placed.

The width of the rectangle is 300 and the height is 200. The definitions for `dot` and `background` are:

```
dot = circle(50, "solid", "red")
background = rectangle(300, 200, "outline", "black")
```

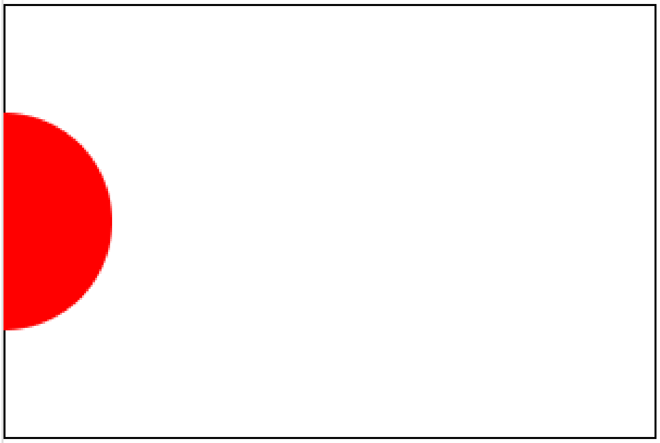
Estimate: What coordinates for the `dot` would create each of the following images?

A



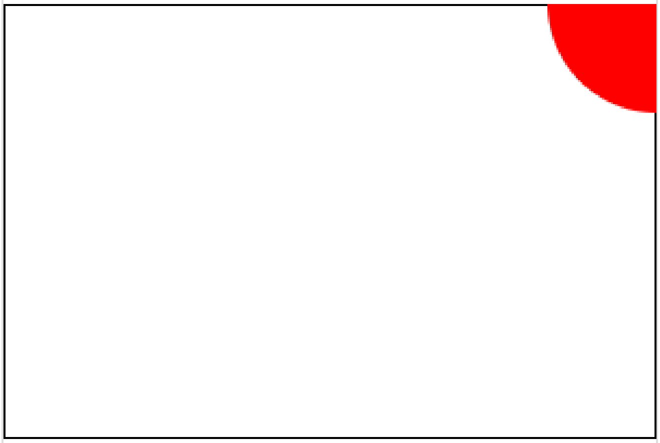
put-image(dot, __________, __________ background)

B



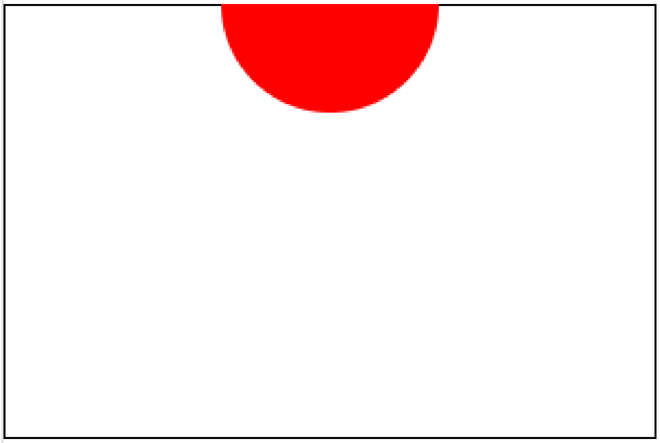
put-image(dot, __________, __________ background)

C



put-image(dot, __________, __________ background)

D

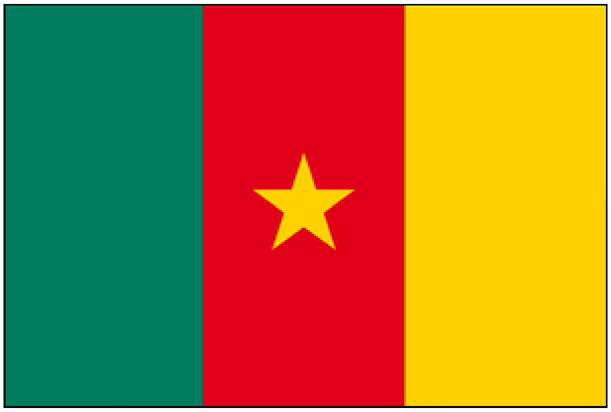


put-image(dot, __________, __________ background)

Decomposing Flags

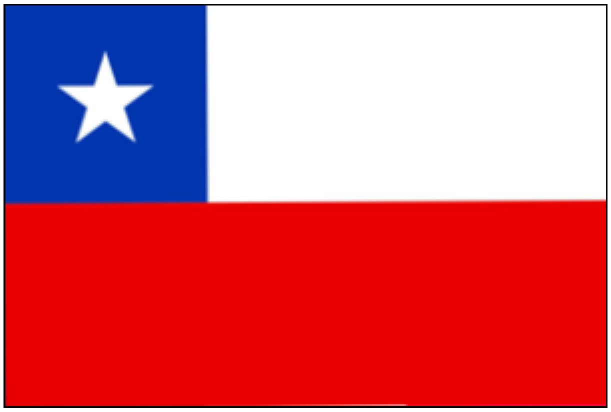
Each of the flags below is shown with their width and height. Identify the shapes that make up each flag. Use the flag's dimensions to estimate the dimensions of the different shapes. Then estimate the x and y coordinates for the point at which the center of each shape should be located on the flag. *Hint: The bottom left corner of each flag is at (0,0) and the top right corner is given by the flags dimensions.*

Cameroon (450 x 300)



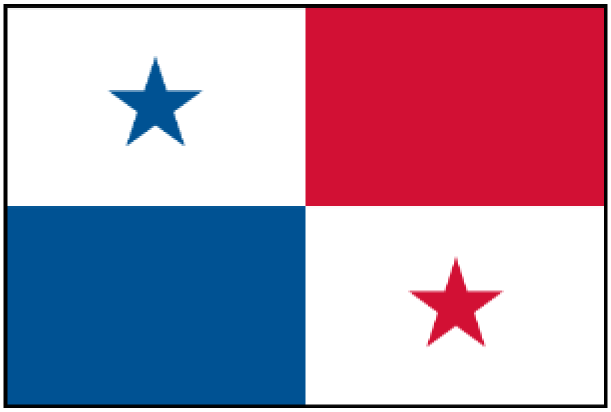
shape:	color:	width:	height:	x	y

Chile (420 x 280)



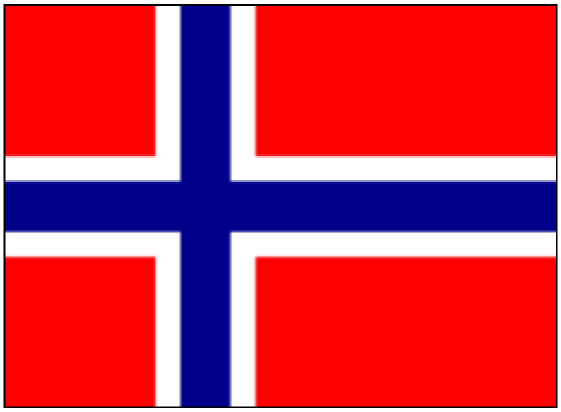
shape:	color:	width:	height:	x	y

Panama (300 x 200)



shape:	color:	width:	height:	x	y

Norway (330 x 240)



shape:	color:	width:	height:	x	y

Defining Functions

Functions can be viewed in *multiple representations*. You already know one of them: ***Contracts***, which specify the Name, Domain, and Range of a function. Contracts are a way of thinking of functions as a *mapping* between one set of data and another. For example, a mapping from Numbers to Strings:

```
f :: Number -> String
```

Another way to view functions is with ***Examples***. Examples are essentially input-output tables, showing what the function would do for a specific input:

In our programming langauge, we focus on the last two columns and write them as code:

```
examples:
  f(1) is 1 + 2
  f(2) is 2 + 2
  f(3) is 3 + 2
  f(4) is 4 + 2
end
```

Finally, we write a formal **function definition** ourselves. The pattern in the Examples becomes *abstract* (or "general"), replacing the inputs with ***variables***. In the example below, the same definition is written in both math and code:

$$f(x) = x + 2$$

```
fun f(x): x + 2 end
```

Look for connections between these three representations!

- The function name is always the same, whether looking at the Contract, Examples, or Definition.
- The number of inputs in the Examples is always the same as the number of types in the Domain, which is always the same as the number of variables in the Definition.
- The "what the function does" pattern in the Examples is almost the same in the Definition, but with specific inputs replaced by variables.

Matching Examples and Definitions (Math)

Look at each set of examples on the left and circle what is changing from one example to the next.
Then, *match* the examples on the left to the definitions on the right.

Examples: **Functions:**

x	$f(x)$
1	2×1
2	2×2
3	2×3

1

A $f(x) = x - 3$

x	$f(x)$
15	$15 - 3$
25	$25 - 3$
35	$35 - 3$

2

B $f(x) = 2x$

x	$f(x)$
10	$10 + 2$
15	$15 + 2$
20	$20 + 2$

3

C $f(x) = 2x + 1$

x	$f(x)$
0	$3(0) - 2$
1	$3(1) - 2$
2	$3(2) - 2$

4

D $f(x) = 3x - 2$

x	$f(x)$
10	$2(10) + 1$
20	$2(20) + 1$
30	$2(30) + 1$

5

E $f(x) = x + 2$

Matching Examples and Function Definitions

Highlight the variables in `gt` and label them with the word "size".

```
examples:
  gt(20) is
    triangle(20, "solid", "green")
  gt(45) is
    triangle(45, "solid", "green")
end
```

```
fun gt(size): triangle(size, "solid", "green") end
```

Highlight and label the variables in the example lists below. Then, using `gt` as a model, match the examples to their corresponding function definitions.

Examples **Definition**

```
examples:
  f("solid") is
    circle(8, "solid", "red")
  f("outline") is
    circle(8, "outline", "red")
end
```

1 A `fun f(s): star(s, "outline", "red") end`

```
examples:
  f(2) is 2 + 2
  f(4) is 4 + 4
  f(5) is 5 + 5
end
```

2 B `fun f(num): num + num end`

```
examples:
  f("red") is circle(7, "solid", "red")
  f("teal") is
    circle(7, "solid", "teal")
end
```

3 C `fun f(c): star(9, "solid", c) end`

```
examples:
  f("red") is star(9, "solid", "red")
  f("grey") is star(9, "solid", "grey")
  f("pink") is star(9, "solid", "pink")
end
```

4 D `fun f(s): circle(8, s, "red") end`

```
examples:
  f(3) is star(3, "outline", "red")
  f(8) is star(8, "outline", "red")
end
```

5 E `fun f(c): circle(7, "solid", c) end`

Matching Examples and Contracts

Match each set of examples (left) with the contract that best describes it(right).

<table>
<tr><th>Examples</th><th></th><th></th><th>Contract</th></tr>
<tr><td><pre>examples:
  f(5) is 5 / 2
  f(9) is 9 / 2
  f(24) is 24 / 2
end</pre></td><td>1</td><td>A</td><td><code># f :: Number -> Number</code></td></tr>
<tr><td><pre>examples:
  f(1) is
    rectangle(1, 1, "outline", "red")
  f(6) is
    rectangle(6, 6, "outline", "red")
end</pre></td><td>2</td><td>B</td><td><code># f :: String -> Image</code></td></tr>
<tr><td><pre>examples:
  f("pink", 5) is
    star(5, "solid", "pink")
  f("blue", 8) is
    star(8, "solid", "blue")
end</pre></td><td>3</td><td>C</td><td><code># f :: Number -> Image</code></td></tr>
<tr><td><pre>examples:
  f("Hi!") is text("Hi!", 50, "red")
  f("Ciao!") is text("Ciao!", 50, "red")
end</pre></td><td>4</td><td>D</td><td><code># f :: Number, String -> Image</code></td></tr>
<tr><td><pre>examples:
  f(5, "outline") is
    star(5, "outline", "yellow")
  f(5, "solid") is
    star(5, "solid", "yellow")
end</pre></td><td>5</td><td>E</td><td><code># f :: String, Number -> Image</code></td></tr>
</table>

Contracts, Examples & Definitions

gt

Directions : Define a function called `gt` , which makes solid green triangles of whatever size we want.

Every contract has three parts...

# gt :: Number -> Image

function name domain range

Write some examples, then circle and label what changes...

examples:

gt (10) **is** triangle(10, "solid", "green")

function name input(s) what the function produces

gt (20) **is** triangle(20, "solid", "green")

function name input(s) what the function produces

end

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

fun gt (size):

function name variable(s)

triangle(size, "solid", "green")

what the function does with those variable(s)

end

bc

Directions : Define a function called `bc` , which makes solid blue circles of whatever radius we want.

Every contract has three parts...

# ______ :: ______ -> ______

function name domain range

Write some examples, then circle and label what changes...

examples:

______ (______) **is** ______

function name input(s) what the function produces

______ (______) **is** ______

function name input(s) what the function produces

end

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

fun ______ (______):

function name variable(s)

______

what the function does with those variable(s)

end

What’s on your mind?

Solving Word Problems

Being able to see functions as Contracts, Examples or Definitions is like having three powerful tools. These representations can be used together to solve word problems!

1) When reading a word problem, the first step is to figure out the **Contract** for the function you want to build. Remember, a Contract must include the Name, Domain and Range for the function!

2) Then we write a **Purpose Statement**, which is a short note that tells us what the function *should do*. Professional programmers work hard to write good purpose statements, so that other people can understand the code they wrote!

3) Next, we write at least two **Examples**. These are lines of code that show what the function should do for a *specific* input. Once we see examples of at least two inputs, we can *find a pattern* and see which parts are changing and which parts aren't.

4) To finish the Examples, we circle the parts that are changing, and label them with a short **variable name** that explains what they do.

5) Finally, we define the function itself! This is pretty easy after you have some examples to work from: we copy everything that didn't change, and replace the changeable stuff with the variable name!

Creating Contracts From Examples

Write the contracts used to create each of the following collections of examples.

1) ____

```
examples:
  big-triangle(100, "red") is
    triangle(100, "solid", "red")
  big-triangle(200, "orange") is
    triangle(200, "solid", "orange")
end
```

2) ____

```
examples:
  purple-square(15) is
    rectangle(15, 15, "outline", "purple")
  purple-square(6) is
    rectangle(6, 6, "outline", "purple")
end
```

3) ____

```
examples:
  banner("Game Today!") is
    text("Game Today!", 50, "red")
  banner("Go Team!") is
    text("Go Team!", 50, "red")
  banner("Exit") is
    text("Exit", 50, "red")
end
```

4) ____

```
examples:
  twinkle("outline", "red") is
    star(5, "outline", "red")
  twinkle("solid", "pink") is
    star(5, "solid", "pink")
  twinkle("outline", "grey") is
    star(5, "outline", "grey")
end
```

5) ____

```
examples:
  half(5) is 5 / 2
  half(8) is 8 / 2
  half(900) is 900 / 2
end
```

Writing Examples from Purpose Statements

We've provided contracts and purpose statements to describe two different functions. Write examples for each of those functions.

Contract and Purpose Statement □

Every contract has three parts…

`upside-down`:: `Image` -> `Image`

function name domain range

Consumes an image, and flips it upside down by rotating it 180 degrees.

what does the function do?

Examples □

Write some examples, then circle and label what changes…

examples:

______ (______) **is**

function name input(s)

______

what the function produces

______ (______) **is**

function name input(s)

______

what the function produces

end

Contract and Purpose Statement □

Every contract has three parts…

`product-squared`:: `Number, Number` -> `Number`

function name domain range

Consumes two numbers and squares their product

what does the function do?

Examples □

Write some examples, then circle and label what changes…

examples:

______ (______) **is** ______

function name input(s) what the function produces

______ (______) **is** ______

function name input(s) what the function produces

end

Word Problem: rocket-height

Directions : A rocket blasts off, and is now traveling at a constant velocity of 7 meters per second. Use the Design Recipe to write a function `rocket-height` , which takes in a number of seconds and calculates the height.

Contract and Purpose Statement □

Every contract has three parts…

____________ :: ______________________________ -> ____________

function name *domain* *range*

____

what does the function do?

Examples □

Write some examples, then circle and label what changes…

```
examples:
  ____________ ( ______________ ) is ____________________
  function name    input(s)          what the function produces
  ____________ ( ______________ ) is ____________________
  function name    input(s)          what the function produces
end
```

Definition □

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

```
fun ____________ ( ______________ ):
    function name    variable(s)

  ______________________________________________
  what the function does with those variable(s)
end
```

Unit 3 (Structures, Reactor, & Animations)

Identifying Animation Data Worksheet

Draw a sketch for three distinct moments of the animation □

Sketch A	Sketch B	Sketch C

What things are changing? □

Thing	Describe how it changes

What fields do you need to represent the things that change? □

Field name (dangerX, score, playerIMG ...)	Data Type (Number, String, Image, Boolean ...)

Data Structure

```
# a
```

Make a sample instance for each sketch from the previous page:

______________________ = ____

______________________ = ____

______________________ = ____

Word Problem: draw-state

Write a function called *draw-state* , which takes in a SunsetState and returns an image in which the sun (a circle) appears at the position given in the SunsetState. The sun should be behing the horizon (the ground) once it is low in the sky.

Contract and Purpose Statement □

```
draw-state :: ______________________________ -> Image
# ____________________________________________________________________
```

Write an expression for each piece of your final image □

SUN =	
GROUND =	
SKY =	

Write the draw-state function, using put-image to combine your pieces □

```
fun _______________(_____________________________________________):
      ____________________________________________________________________
      ____________________________________________________________________
      ____________________________________________________________________
end
```

Word Problem: next-state-tick

Directions : Write a function called *next-state-tick* , which takes in a SunsetState and returns a SunsetState in which the new x-coordinate is 8 pixels larger than in the given SunsetState and the y-coordinate is 4 pixels smaller than in the given SunsetState.

Contract and Purpose Statement □

Every contract has three parts…

________________ :: ____________________________________ -> ________________

function name *domain* *range*

____

what does the function do?

Examples □

Write some examples, then circle and label what changes…

examples:

________________ (____________________) **is** __________________________________

function name *input(s)* *what the function produces*

________________ (____________________) **is** __________________________________

function name *input(s)* *what the function produces*

end

Definition □

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

fun ________________ (____________________):

function name *variable(s)*

____

what the function does with those variable(s)

end

Identifying Animation Data Worksheet

Draw a sketch for three distinct moments of the animation □

Sketch A	Sketch B	Sketch C

What things are changing? □

Thing	Describe how it changes

What fields do you need to represent the things that change? □

Field name (dangerX, score, playerIMG …)	Data Type (Number, String, Image, Boolean …)

Data Structure

```
# a
```

Make a sample instance for each sketch from the previous page:

____________________ = ____

____________________ = ____

____________________ = ____

Identifying Animation Data Worksheet

Draw a sketch for the three distinct moments of the animation □

Sketch A	Sketch B	Sketch C

What things are changing? □

Thing	Describe how it changes

What fields do you need to represent the things that change? □

Field name (dangerX, score, playerIMG ...)	Data Type (Number, String, Image, Boolean ...)

Data Structure

```
# a
```

Make a sample instance for each sketch from the previous page:

______________________ = ____

______________________ = ____

______________________ = ____

Identifying Animation Data Worksheet

Draw a sketch for three distinct moments of the animation □

Sketch A	Sketch B	Sketch C

What things are changing? □

Thing	Describe how it changes

What fields do you need to represent the things that change? □

Field name (dangerX, score, playerIMG ...)	Data Type (Number, String, Image, Boolean ...)

Data Structure

```
# a
```

Make a sample instance for each sketch from the previous page:

______________________ = ____

______________________ = ____

______________________ = ____

Identifying Animation Data Worksheet

Draw a sketch for three distinct moments of the animation □

Sketch A	Sketch B	Sketch C

What things are changing? □

Thing	Describe how it changes

What fields do you need to represent the things that change? □

Field name (dangerX, score, playerIMG ...)	Data Type (Number, String, Image, Boolean ...)

Data Structure

```
# a
```

Make a sample instance for each sketch from the previous page:

________________________ = ____

________________________ = ____

________________________ = ____

Unit 4 (Functions That Ask Questions)

Word Problem: location

Directions : Write a function called location, which consumes a DeliveryState, and produces a String representing the location of a box: either "road", "delivery zone", "house", or "air".

Contract and Purpose Statement □

Every contract has three parts…

# ________________ :: ____ -> ________________

function name *domain* *range*

# ____

what does the function do?

Examples □

Write some examples, then circle and label what changes…

examples:

____________ (________________) **is** __________________________

function name *input(s)* *what the function produces*

____________ (________________) **is** __________________________

function name *input(s)* *what the function produces*

____________ (________________) **is** __________________________

function name *input(s)* *what the function produces*

____________ (________________) **is** __________________________

function name *input(s)* *what the function produces*

end

Definition □

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

fun ____________ (________________):

function name *variable(s)*

____

what the function does with those variable(s)

end

Syntax and Style Bug Hunting: Piecewise Edition

	Buggy Code	Correct Code / Explanation

1	`fun piecewisefun(n):`
	` if (n > 0): n`
	` else: 0`	

2	`fun cost(topping):`
	` if string-equal(topping, "pepperoni"): 10.50`
	` else string-equal(topping, "cheese"): 9.00`
	` else string-equal(topping, "chicken"): 11.25`
	` else string-equal(topping, "broccoli"): 10.25`
	` else: "That's not on the menu!"`
	` end`
	`end`	

3	`fun absolute-value(a b):`
	` if a > b: a - b`
	` b - a`
	` end`
	`end`	

4	`fun best-function(f):`
	` if string-equal(f, “blue”): “you win!”`
	` else if string-equal(f, “blue”): “you lose!”`
	` else if string-equal(f, “red”): “Try again!”`
	` else: “Invalid entry!”`
	` end`
	`end`	

Animation Data Worksheet

Decrease the cat's hunger level by 2 and sleep level by 1 on each tick.

Draw a sketch for three distinct moments of the animation □

Sketch A	**Sketch B**	**Sketch C**

What things are changing? □

Thing	Describe how it changes

What fields do you need to represent the things that change? □

Field name (dangerX, score, playerIMG ...)	data type (Number, String, Image, Boolean ...)

Make a To-Do List, and check off each as "Done" when you finish each one. □

Component	*When is there work to be done?*	To-Do	Done
Data Structure	*If any new field(s) were added, changed, or removed*	□	□
draw-state	*If something is displayed in a new way or position*	☑	□
next-state-tick	*If the Data Structure changed, or the animation happens automatically*	□	□
next-state-key	*If the Data Structure changed, or a keypress triggers the animation*	□	□
reactor	*If either next-state function is new*	□	□

1) Make a sample instance for each sketch from the previous page:

_______________ =

____

_______________ =

____

_______________ =

____

2) Write at least one NEW example for one of the functions on your To-Do list

3) If you have another function on your To-Do list, write at least one NEW example

Word Problem: draw-sun

Directions : Write a function called draw-sun, which consumes a SunsetState, and produces an image of a sun (a solid, 25 pixel circle), whose color is "yellow", when the sun’s y-coordinate is greater than 225, "orange", when its y-coordinate is between 150 and 225, and "red" otherwise.

Contract and Purpose Statement □

Every contract has three parts…

# ______________ :: ______________________________________ -> ______________

function name *domain* *range*

# ____

what does the function do?

Examples □

Write some examples, then circle and label what changes…

`examples:`

______________ (______________) **is** ______________________

function name *input(s)* *what the function produces*

______________ (______________) **is** ______________________

function name *input(s)* *what the function produces*

______________ (______________) **is** ______________________

function name *input(s)* *what the function produces*

______________ (______________) **is** ______________________

function name *input(s)* *what the function produces*

`end`

Definition □

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

`fun` ______________ (______________):

function name *variable(s)*

____

what the function does with those variable(s)

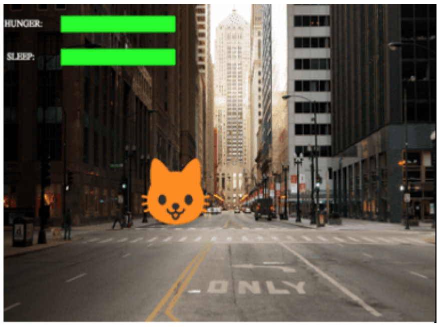
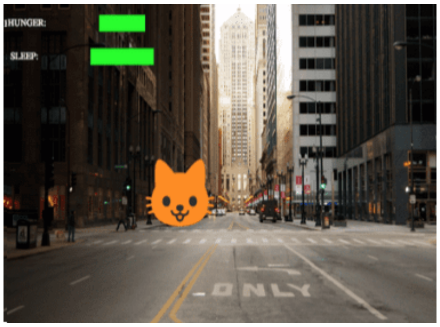
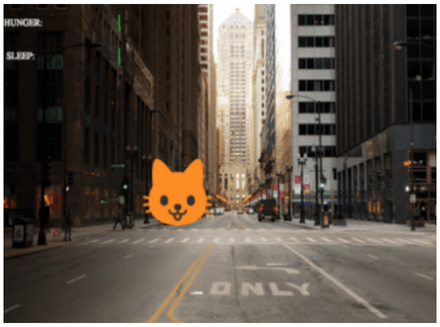
`end`

Unit 5 (Key Events)

Animation Data Worksheet

Decrease the cat’s hunger level by 2 and sleep level by 1 on each tick.

Draw a sketch for three distinct moments of the animation □

		
Sketch A	Sketch B	Sketch C

What things are changing? □

Thing	Describe how it changes

What fields do you need to represent the things that change? □

Field name (dangerX, score, playerIMG ...)	data type (Number, String, Image, Boolean ...)

Make a To-Do List, and check off each as “Done” when you finish each one. □

Component	*When is there work to be done?*	To-Do	Done
Data Structure	*If any new field(s) were added, changed, or removed*	□	□
draw-state	*If something is displayed in a new way or position*	☑	□
next-state-tick	*If the Data Structure changed, or the animation happens automatically*	□	□
next-state-key	*If the Data Structure changed, or a keypress triggers the animation*	□	□
reactor	*If either next-state function is new*	□	□

1) Make a sample instance for each sketch from the previous page:

FULLPET =

pet(100, 100)

MIDPET =

pet(50, 75)

LOSEPET =

pet(0, 0)

2) Write at least one NEW example for one of the functions on your To-Do list

next-state-tick(FULLPET) is pet(FULLPET.hunger – 2, FULLPET.sleep – 1)

next-state-tick(MIDPET) is pet(MIDPET.hunger – 2, MIDPET.sleep – 1)

next-state-tick(LOSEPET) is LOSEPET

3) If you have another function on your To-Do list, write at least one NEW example

Animation Data Worksheet

Decrease the cat’s hunger level by 2 and sleep level by 1 on each tick.

Draw a sketch for three distinct moments of the animation □

Sketch A	**Sketch B**	**Sketch C**

What things are changing? □

Thing	Describe how it changes

What fields do you need to represent the things that change? □

Field name (dangerX, score, playerIMG ...)	data type (Number, String, Image, Boolean ...)

Make a To-Do List, and check off each as “Done” when you finish each one. □

Component	*When is there work to be done?*	To-Do	Done
Data Structure	*If any new field(s) were added, changed, or removed*	□	□
draw-state	*If something is displayed in a new way or position*	☑	□
next-state-tick	*If the Data Structure changed, or the animation happens automatically*	□	□
next-state-key	*If the Data Structure changed, or a keypress triggers the animation*	□	□
reactor	*If either next-state function is new*	□	□

1) Make a sample instance for each sketch from the previous page:

_____________ =

____

_____________ =

____

_____________ =

____

2) Write at least one NEW example for one of the functions on your To-Do list

____

____

____

____

____

3) If you have another function on your To-Do list, write at least one NEW example

____

____

____

____

____

Animation Data Worksheet

Decrease the cat’s hunger level by 2 and sleep level by 1 on each tick.

Draw a sketch for three distinct moments of the animation □

Sketch A	**Sketch B**	**Sketch C**

What things are changing? □

Thing	Describe how it changes

What fields do you need to represent the things that change? □

Field name (dangerX, score, playerIMG …)	data type (Number, String, Image, Boolean …)

Make a To-Do List, and check off each as “Done” when you finish each one. □

Component	*When is there work to be done?*	To-Do	Done
Data Structure	*If any new field(s) were added, changed, or removed*	□	□
draw-state	*If something is displayed in a new way or position*	☑	□
next-state-tick	*If the Data Structure changed, or the animation happens automatically*	□	□
next-state-key	*If the Data Structure changed, or a keypress triggers the animation*	□	□
reactor	*If either next-state function is new*	□	□

1) Make a sample instance for each sketch from the previous page:

______________ =

____

______________ =

____

______________ =

____

2) Write at least one NEW example for one of the functions on your To-Do list

____

____

____

____

____

3) If you have another function on your To-Do list, write at least one NEW example

____

____

____

____

____

Refactoring

Your Own Drawing Functions

Build Your Own Animation

Animation Data Worksheet

Decrease the cat’s hunger level by 2 and sleep level by 1 on each tick.

Draw a sketch for three distinct moments of the animation □

Sketch A	Sketch B	Sketch C

What things are changing? □

Thing	Describe how it changes

What fields do you need to represent the things that change? □

Field name (dangerX, score, playerIMG ...)	data type (Number, String, Image, Boolean ...)

Make a To-Do List, and check off each as “Done” when you finish each one. □

Component	*When is there work to be done?*	To-Do	Done
Data Structure	*If any new field(s) were added, changed, or removed*	□	□
draw-state	*If something is displayed in a new way or position*	☑	□
next-state-tick	*If the Data Structure changed, or the animation happens automatically*	□	□
next-state-key	*If the Data Structure changed, or a keypress triggers the animation*	□	□
reactor	*If either next-state function is new*	□	□

Define the Data Structure □

```
# a ______________State is ______________________________________________
data ______________State:
| ______________ (_______________________________________________________________________
                    _______________________________________________________________________
                    _______________________________________________________________________
                    _______________________________________________________________________)
end
```

Make a sample instance for each sketch from the previous page □

```
______________ = _______________________________________________________________________
______________ = _______________________________________________________________________
______________ = _______________________________________________________________________
```

Write an example for one of the functions on the previous page □

____

____

____

____

____

Collisions

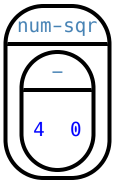
Distance

The Player is at (4, 2) and the Target is at (0, 5).

Distance takes in the player's x, player's y, character's x and character's y. Use the formula below to fill in the EXAMPLE:

$$\sqrt{(4-0)^2 + (2-5)^2}$$

Convert it into a Circle of Evaluation. (We've already gotten you started!)



Convert it to Pyret code.

Word Problem: distance

Directions : Write a function `distance` , which takes FOUR inputs: (1) px: The x-coordinate of the player, (2) py: The y-coordinate of the player, (3) cx: The x-coordinate of another game character, (4) cy: The y-coordinate of another game character. It should return the distance between the two, using the Distance formula: Distance^2 = (px − cx)^2 + (py − cy)^2

Contract and Purpose Statement □

Every contract has three parts…

________________ :: ____ -> ________________

function name *domain* *range*

____

what does the function do?

Examples □

Write some examples, then circle and label what changes…

examples:

________________ (________________) **is** ____________________________

function name *input(s)* *what the function produces*

________________ (________________) **is** ____________________________

function name *input(s)* *what the function produces*

end

Definition □

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

fun ________________ (________________):

function name *variable(s)*

____

what the function does with those variable(s)

end

Word Problem: is-collision

Directions : Write a function `is-collision` , which takes FOUR inputs: (1) px: The x-coordinate of the player, (2) py: The y-coordinate of the player, (3) cx: The x-coordinate of another game character, (4) cy: The y-coordinate of another game character. It should return true if the coordinates of the player are within **50 pixels** of the coordinates of the other character. Otherwise, false.

Contract and Purpose Statement □

Every contract has three parts...

# ________________ :: ____ -> ________________

function name domain range

# ____

what does the function do?

Examples □

Write some examples, then circle and label what changes...

examples:

________________ (____________________) **is** ______________________________

function name input(s) what the function produces

________________ (____________________) **is** ______________________________

function name input(s) what the function produces

end

Definition □

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

fun ________________ (____________________):

function name variable(s)

____

what the function does with those variable(s)

end

Notes

Making Pong

Nested Structures

Timers

Directions : sp

sp

sp

Contract and Purpose Statement □

Every contract has three parts…

________________ :: ____ -> ________________

function name *domain* *range*

____

what does the function do?

Examples □

Write some examples, then circle and label what changes…

examples:

________________ (________________) **is** ____________________________

function name *input(s)* *what the function produces*

________________ (________________) **is** ____________________________

function name *input(s)* *what the function produces*

end

Definition □

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

fun ________________ (________________):

function name *variable(s)*

____

what the function does with those variable(s)

end

Directions : sp

sp

sp

Contract and Purpose Statement □

Every contract has three parts…

________________ :: ____ -> ________________

function name *domain* *range*

____

what does the function do?

Examples □

Write some examples, then circle and label what changes…

```
examples:
```

________________ (____________________) **is** ________________________________

function name *input(s)* *what the function produces*

________________ (____________________) **is** ________________________________

function name *input(s)* *what the function produces*

```
end
```

Definition □

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

fun ________________ (____________________):

function name *variable(s)*

____

what the function does with those variable(s)

```
end
```

Animation Data Worksheet

Decrease the cat's hunger level by 2 and sleep level by 1 on each tick.

Draw a sketch for three distinct moments of the animation □

Sketch A	Sketch B	Sketch C

What things are changing? □

Thing	Describe how it changes

What fields do you need to represent the things that change? □

Field name (dangerX, score, playerIMG ...)	Datatype (Number, String, Image, Boolean ...)

Make a To-Do List, and check off each as "Done" when you finish each one. □

Component	*When is there work to be done?*	To-Do	Done
Data Structure	*If any new field(s) were added, changed, or removed*	□	□
draw-state	*If something is displayed in a new way or position*	☑	□
next-state-tick	*If the Data Structure changed, or the animation happens automatically*	□	□
next-state-key	*If the Data Structure changed, or a keypress triggers the animation*	□	□
reactor	*If either next-state function is new*	□	□

Define the Data Structure □

```
# a ______________State is _______________________________________________
data ______________State:
| ______________ (______________________________________________________________________
                   ______________________________________________________________________
                   ______________________________________________________________________
                   ______________________________________________________________________)
end
```

Make a sample instance for each sketch from the previous page □

```
______________ = ______________________________________________________________________
______________ = ______________________________________________________________________
______________ = ______________________________________________________________________
```

Write an example for one of the functions on the previous page □

____

____

____

____

____

Animation Data Worksheet

Decrease the cat's hunger level by 2 and sleep level by 1 on each tick.

Draw a sketch for three distinct moments of the animation □

Sketch A	**Sketch B**	**Sketch C**

What things are changing? □

Thing	Describe how it changes

What fields do you need to represent the things that change? □

Field name (dangerX, score, playerIMG …)	Datatype (Number, String, Image, Boolean …)

Make a To-Do List, and check off each as "Done" when you finish each one. □

Component	*When is there work to be done?*	To-Do	Done
Data Structure	*If any new field(s) were added, changed, or removed*	□	□
draw-state	*If something is displayed in a new way or position*	☑	□
next-state-tick	*If the Data Structure changed, or the animation happens automatically*	□	□
next-state-key	*If the Data Structure changed, or a keypress triggers the animation*	□	□
reactor	*If either next-state function is new*	□	□

Define the Data Structure □

```
# a ______________State is ______________________________________________
data ______________State:
| ______________ (______________________________________________________
                    ______________________________________________________
                    ______________________________________________________
                    ______________________________________________________)
end
```

Make a sample instance for each sketch from the previous page □

```
______________ = ______________________________________________________
______________ = ______________________________________________________
______________ = ______________________________________________________
```

Write an example for one of the functions on the previous page □

____

____

____

____

____

Animation Data Worksheet

Decrease the cat’s hunger level by 2 and sleep level by 1 on each tick.

Draw a sketch for three distinct moments of the animation □

Sketch A	Sketch B	Sketch C

What things are changing? □

Thing	Describe how it changes

What fields do you need to represent the things that change? □

Field name (dangerX, score, playerIMG …)	Datatype (Number, String, Image, Boolean …)

Make a To-Do List, and check off each as “Done” when you finish each one. □

Component	*When is there work to be done?*	To-Do	Done
Data Structure	*If any new field(s) were added, changed, or removed*	□	□
draw-state	*If something is displayed in a new way or position*	☑	□
next-state-tick	*If the Data Structure changed, or the animation happens automatically*	□	□
next-state-key	*If the Data Structure changed, or a keypress triggers the animation*	□	□
reactor	*If either next-state function is new*	□	□

Define the Data Structure □

```
# a ______________State is ______________________________________________
data ______________State:
| ______________ (______________________________________________________
                   ______________________________________________________
                   ______________________________________________________
                   ______________________________________________________)
end
```

Make a sample instance for each sketch from the previous page □

```
______________ = ______________________________________________________
______________ = ______________________________________________________
______________ = ______________________________________________________
```

Write an example for one of the functions on the previous page □

____

____

____

____

____

Contracts

Contracts tell us how to use a function. For example: `num-sqr two-colons (n two-colons Number) -> Number` tells us that the name of the function is `num-sqr` , it takes one input (a `Number`), and it evaluates to a `Number` . From the contract, we know `num-sqr(4)` will evaluate to a `Number` .

Name		Domain		Range
# triangle	::	(side-length :: Number, style :: String, color :: String)	->	Image
#				
# circle	::	(radius :: Number, style :: String, color :: String)	->	Image
#				
# star	::	(radius :: Number, style :: String, color :: String)	->	Image
#				
# rectangle	::	(width :: Num, height :: Num, style :: Str, color :: Str)	->	Image
#				
# ellipse	::	(width :: Num, height :: Num, style :: Str, color :: Str)	->	Image
#				
# square	::	(size-length :: Number, style :: String, color :: String)	->	Image
#				
# text	::	(str :: String, size :: Number, color :: String)	->	Image
#				
# overlay	::	(img1 :: Image, img2 :: Image)	->	Image
#				
# beside	::	(img1 :: Image, img2 :: Image)	->	Image
#				
# image-url	::	(url :: String)	->	Image
#				

Contracts

Contracts tell us how to use a function. For example: `num-sqr :: (n :: Number) -> Number` tells us that the name of the function is `num-sqr`, it takes one input (a `Number`), and it evaluates to a `Number`. From the contract, we know `num-sqr(4)` will evaluate to a `Number`.

Name		Domain		Range
`# above`	`::`	`(img1 :: Image, img2 :: Image)`	`->`	`Image`
`#`				
`# put-image`	`::`	`(img1 :: Image, x :: Number, y :: Number, img2 :: Image)`	`->`	`Image`
`#`				
`# rotate`	`::`	`(degree :: Number, img :: Image)`	`->`	`Image`
`#`				
`# scale`	`::`	`(factor :: Number, img :: Image)`	`->`	`Image`
`#`				
`# string-repeat`	`::`	`(text :: String, repeat :: Number)`	`->`	`String`
`#`				
`# string-contains`	`::`	`(text :: String, search-for :: String)`	`->`	`Boolean`
`#`				
`# num-sqr`	`::`	`(n :: Number)`	`->`	`Number`
`#`				
`# num-sqrt`	`::`	`(n :: Number)`	`->`	`Number`
`#`				
`# num-min`	`::`	`(a :: Number, b:: Number)`	`->`	`Number`
`#`				
`# num-max`	`::`	`(a :: Number, b:: Number)`	`->`	`Number`
`#`				

Contracts

Contracts tell us how to use a function. For example: `num-sqr :: (n :: Number) -> Number` tells us that the name of the function is `num-sqr`, it takes one input (a `Number`), and it evaluates to a `Number`. From the contract, we know `num-sqr(4)` will evaluate to a `Number`.

Name		Domain		Range
# string-equal	::	(str1 :: String, str2 :: String)	->	Boolean
#				
# and	::	(test1 :: Boolean, test2 :: Boolean)	->	Boolean
#				
# or	::	(test1 :: Boolean, test2 :: Boolean)	->	Boolean
#				
#	::		->	
#				
#	::		->	
#				
#	::		->	
#				
#	::		->	
#				
#	::		->	
#				
#	::		->	
#				
#	::		->	
#				

9 781365 212666

These materials were developed partly through support of the National Science Foundation, (awards 1042210, 1535276, 1648684, and 1738598), and are licensed under a Creative Commons 4.0 Unported License. Based on a work at www.BootstrapWorld.org. Permissions beyond the scope of this license may be available by contacting schanzer@BootstrapWorld.org.