Fill out the contract for each function, then try to write two examples and the definition by yourself.

Directions:

**Contract and Purpose Statement**
Every contract has three parts…

# double :: __________________ --> __________________
# __________________

what does the function do?

**Examples**
Write some examples, then circle and label what changes…

eamples:

```
double ( 5 ) is 2 * 5
```

what the function produces

```
double ( 7 ) is 2 * 7
```

what the function produces

end

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

```fun
double ( n ): 2 * n
```

what the function does with those variable(s)
Fast Functions!

Fill out the contract for each function, then try to write two examples and the definition by yourself.

Directions:

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…

eamples:

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

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

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

fun (variable(s)):

function name variable(s)

what the function does with those variable(s)

Directions:

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…

eamples:

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

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

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

fun (variable(s)):

function name variable(s)

what the function does with those variable(s)
Fast Functions!

Fill out the contract for each function, then try to write two examples and the definition by yourself.

Directions:

**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:

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

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

end```

**Definition**

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

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

function name variable(s)

what the function does with those variable(s)

end```

Directions:

**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:

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

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

end```

**Definition**

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

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

function name variable(s)

what the function does with those variable(s)

end```
Word Problem: double-radius

Directions: Write a function *double-radius*, which takes in a radius and a color. It produces an outlined circle of whatever color was passed in, whose radius is twice as big as the input.

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

Don't care
Word Problem: double-radius

Directions: Write a function *double-width*, which takes in a number (the length of a rectangle) and produces a rectangle whose length is twice the given length.

### 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:

```
function name (input(s)) is what the function produces
```

**end**

Don't care
Word Problem: next-position

Directions: Write a function next-position, which takes in two numbers (an x- and y-coordinate) and returns a DeliveryState, increasing the x-coordinate by 5 and decreasing the y-coordinate by 5.

Contract and Purpose Statement
Every contract has three parts…

# _______________ :: _______________ -> _______________

# _______________ 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

Don't care
Data Structure

data CakeType:
  | cake(____________________________________________________
  | ______________________________________________________
  | ______________________________________________________
  | ______________________________________________________
end

to make an instance of this structure, I would write:

cake1 = __________________________________________________
cake2 = __________________________________________________

to access the fields of cake2, I would write:

____________________________________________________
____________________________________________________
____________________________________________________
Word Problem: taller-than

Directions: Write a function called taller-than, which consumes two CakeTypes, and produces true if the number of layers in the first CakeType is greater than the number of layers in the second.

Contract and Purpose Statement

Every contract has three parts…

```
# function name :: domain -> range
```

Examples

Write some examples, then circle and label what changes…

examples:

```
function name (input(s)) is what the function produces
```

end

Don't care
**Word Problem: will-melt**

**Directions:** Write a function called `will-melt`, which takes in a CakeType and a temperature, and returns true if the temperature is greater than 32 degrees, AND the CakeType is an ice-cream cake.

**Contract and Purpose Statement**

Every contract has three parts…

# ________________ :: ____________________ -> ____________________

# ____________________________

what does the function do?

**Examples**

Write some examples, then circle and label what changes…

examples:

_________ (___________) is _________________

_________ (___________) is _________________

end

Don't care
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 behind 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**

<table>
<thead>
<tr>
<th>SUN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GROUND</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SKY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

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

```latex
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…*

# __________________ : ____________________ -> __________________

<table>
<thead>
<tr>
<th>function name</th>
<th>domain</th>
<th>range</th>
</tr>
</thead>
</table>

# __________________

what does the function do?

**Examples**

*Write some examples, then circle and label what changes…*

**examples:**

<table>
<thead>
<tr>
<th>function name</th>
<th>input(s)</th>
<th>what the function produces</th>
</tr>
</thead>
</table>

**end**

Don’t care
## Identifying Animation Data Worksheet: Sunset

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

<table>
<thead>
<tr>
<th>Sketch A</th>
<th>Sketch B</th>
<th>Sketch C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### What things are changing?

<table>
<thead>
<tr>
<th>Thing</th>
<th>Describe how it changes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Field name (dangerX, score, playerIMG …)</th>
<th>Datatype (Number, String, Image, Boolean …)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Animation Data Worksheet**

Draw a sketch for three distinct moments of the animation

<table>
<thead>
<tr>
<th>Sketch A</th>
<th>Sketch B</th>
<th>Sketch C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What things are changing?

<table>
<thead>
<tr>
<th>Thing</th>
<th>Describe how it changes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Field name (dangerX, score, playerIMG ...)</th>
<th>Datatype (Number, String, Image, Boolean ...)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Component</th>
<th>When is there work to be done?</th>
<th>To-Do</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Structure</td>
<td>_If any new field(s) were added, changed, or removed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>draw-state</td>
<td>_If something is displayed in a new way or position</td>
<td></td>
<td></td>
</tr>
<tr>
<td>next-state-tick</td>
<td>_If the Data Structure changed, or the animation happens automatically</td>
<td></td>
<td></td>
</tr>
<tr>
<td>next-state-key</td>
<td>_If the Data Structure changed, or a keypress triggers the animation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>reactor</td>
<td>_If either next-state function is new</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

__________________________________________________________________________________
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:
\[(4 - 0)^2 + (2 - 5)^2\]
Convert it into a Circle of Evaluation. (We've already gotten you started!)

\[
\begin{array}{c}
\text{num-sqr} \\
- \\
4 \quad 0
\end{array}
\]

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: \( \text{Distance}^2 = (px - cx)^2 + (py - cy)^2 \)

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:

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

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

end

Don't care
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 |

g______________ (________________) is ______________
  | function name | input(s) | what the function produces |

end

Don't care
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`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td># triangle</td>
<td>(side-length :: Number, style :: String, color :: String)</td>
<td>➞ Image</td>
</tr>
<tr>
<td># circle</td>
<td>(radius :: Number, style :: String, color :: String)</td>
<td>➞ Image</td>
</tr>
<tr>
<td># star</td>
<td>(radius :: Number, style :: String, color :: String)</td>
<td>➞ Image</td>
</tr>
<tr>
<td># rectangle</td>
<td>(width :: Num, height :: Num, style :: Str, color :: Str)</td>
<td>➞ Image</td>
</tr>
<tr>
<td># ellipse</td>
<td>(width :: Num, height :: Num, style :: Str, color :: Str)</td>
<td>➞ Image</td>
</tr>
<tr>
<td># square</td>
<td>(size-length :: Number, style :: String, color :: String)</td>
<td>➞ Image</td>
</tr>
<tr>
<td># text</td>
<td>(str :: String, size :: Number, color :: String)</td>
<td>➞ Image</td>
</tr>
<tr>
<td># overlay</td>
<td>(img1 :: Image, img2 :: Image)</td>
<td>➞ Image</td>
</tr>
<tr>
<td># beside</td>
<td>(img1 :: Image, img2 :: Image)</td>
<td>➞ Image</td>
</tr>
<tr>
<td># bitmap-url</td>
<td>(url :: String)</td>
<td>➞ Image</td>
</tr>
</tbody>
</table>
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`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td># above</td>
<td>::</td>
<td></td>
</tr>
<tr>
<td># put-image</td>
<td>::</td>
<td></td>
</tr>
<tr>
<td># rotate</td>
<td>::</td>
<td></td>
</tr>
<tr>
<td># scale</td>
<td>::</td>
<td></td>
</tr>
<tr>
<td># string-repeat</td>
<td>::</td>
<td></td>
</tr>
<tr>
<td># string-contains</td>
<td>::</td>
<td></td>
</tr>
<tr>
<td># num-sqr</td>
<td>::</td>
<td></td>
</tr>
<tr>
<td># num-sqrt</td>
<td>::</td>
<td></td>
</tr>
<tr>
<td># num-min</td>
<td>::</td>
<td></td>
</tr>
<tr>
<td># num-max</td>
<td>::</td>
<td></td>
</tr>
</tbody>
</table>
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`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td># string-equal</td>
<td>:: (str1 :: String, str2 :: String)</td>
<td>-&gt; Boolean</td>
</tr>
<tr>
<td># and</td>
<td>:: (test1 :: Boolean, test2 :: Boolean)</td>
<td>-&gt; Boolean</td>
</tr>
<tr>
<td># or</td>
<td>:: (test1 :: Boolean, test2 :: Boolean)</td>
<td>-&gt; Boolean</td>
</tr>
</tbody>
</table>