Mission 2: Rune Reading

Start date: 25 August 2017
Due: 29 August 2017, 23:59

Readings:

- Textbook Sections 1.1.1 to 1.1.4

In Mission 1, we have tried our hands at drawing some simple patterns. Using what we have learnt about recursion in the past few days, we are going to practice drawing more complex patterns!

For this mission, the Playground loads the `rune_2d` script library. This provides all the runes and relevant methods mentioned in the lecture slides and mission 1. Please refer to these materials if you want to know the specifications of these methods.

One useful method for Tasks 2 and 3 will be stack_frac, which has the following specification:

```
stack_frac(frac, picture1, picture2)
--> returns an image where picture1 is stacked on top of picture2
    and picture1 occupies the top frac of the image, while picture2
    occupies the remaining 1 - frac of the image
```

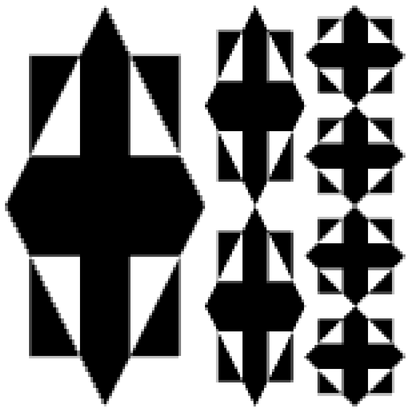
This mission consists of **three** tasks. **Click here for the link to the template**. Use the template provided in the link to complete the tasks.

Task 1:

Write a function `fractal` that takes as arguments a rune and an integer n > 0. It should generate the rune below by means of a recursive process with the following command:

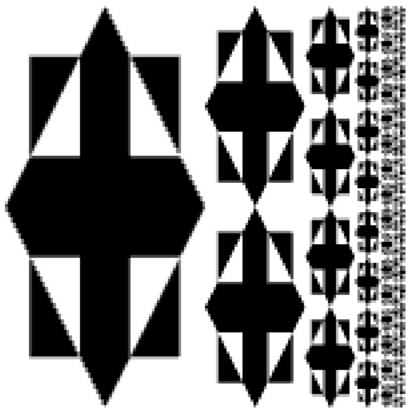
```
show(fractal(make_cross(rcross_bb), 3));
```

This should draw the following:



To determine that your function is correct for n > 3, check that
the same command with n = 7 draws, i.e.:

```
show(fractal(make_cross(rcross_bb), 7));
```

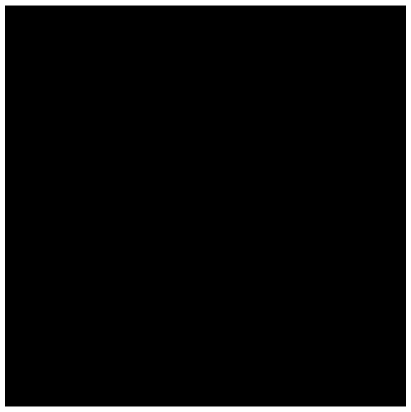


Task 2:

Write a function `hook` that takes a fraction as an input, and creates a 'hook' pattern. The fraction input determines the size of the base of the hook. We shall see a few examples to show what we want and highlight how changing the input modifies the picture.

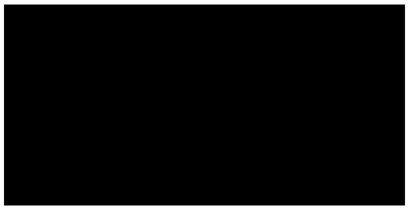
```
show(hook(1));
```

should draw the following:



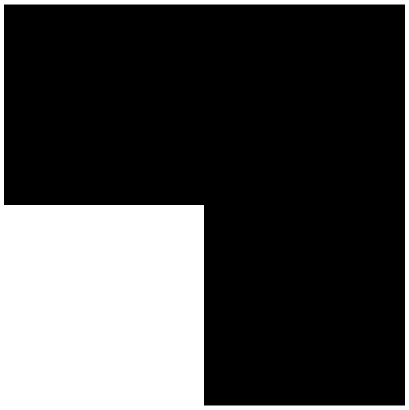
```
show(hook(0));
```

should draw the following:



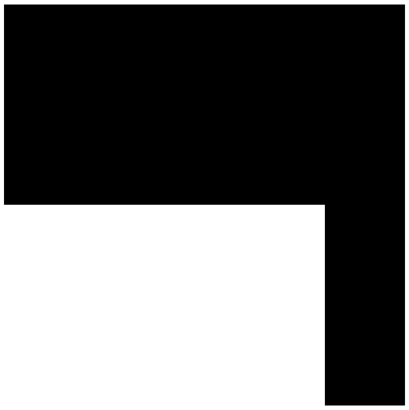
```
show(hook(1/2));
```

should draw the following:



```
show(hook(1/5));
```

should draw the following:



HINT: You will only need to use the black_bb and blank_bb images and transform them to get the hook. You will also need to make use of the stack_frac function.

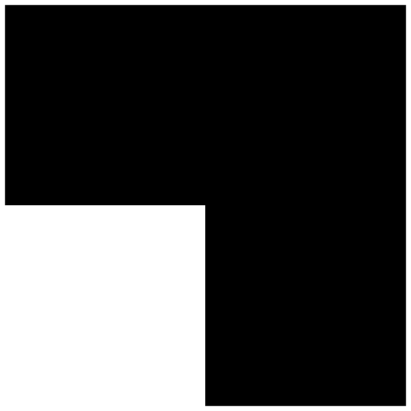
Task 3:

Write a function `spiral` that takes a fraction "gradient" and a integer "n" > 0 as inputs and creates a spiral through recursion. The gradient determines the thickness of the spiral, while the integer determines the depth of the spiral. We shall see a few examples to show what we want and highlight how changing the inputs modifies the picture.

NOTE: If you have written your program correctly, spiral should also return an empty image if gradient = 0 or n = 0.

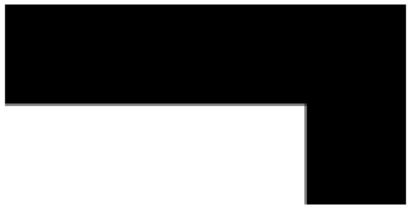
`show(spiral(1, 1));`

should draw the following:



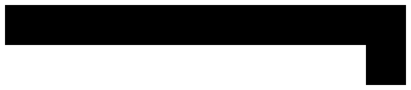
`show(spiral(1/2, 1));`

should draw the following:



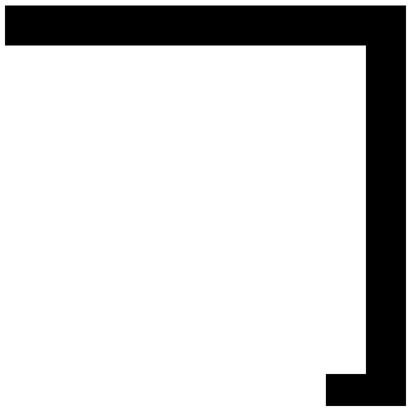
`show(spiral(1/5, 1));`

should draw the following:



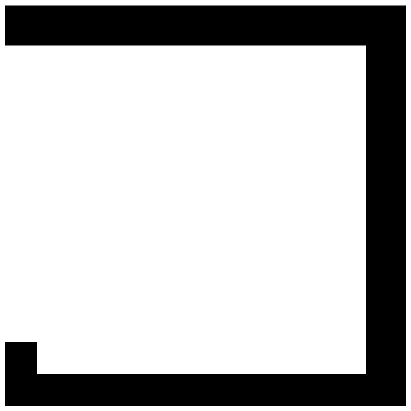
show(spiral(1/5, 2));

should draw the following:



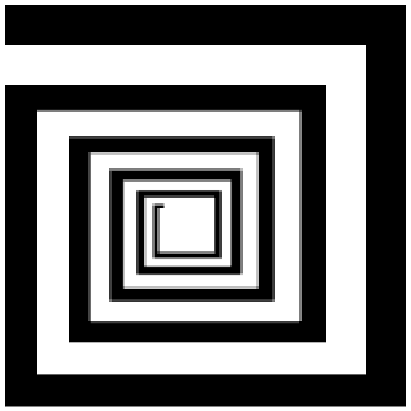
show(spiral(1/5, 3));

should draw the following:



show(spiral(1/5, 20));

should draw the following:



HINT: Make use of a ‘hook’ as the basic building block of the spiral and consider whether you should apply hook to gradient or a fraction of gradient. Lastly, think about how to create a spiral of depth 1 using a hook and how to use a spiral of depth 1 to create a spiral of level 2 and so on.

Submission

To submit your work for this mission, copy the url on your browser and email it to your respective Avengers. Strictly follow to the deadlines set at the start of this file.
IMPORTANT: Make sure that everything for your programs to work is on the left hand side (source code) and not in the interpreter! This is because only that code is preserved when opening the url you have emailed to us.