

COMPUTER SCIENCE

LAB 9

BOOLEANS AND CONDITIONALS

For each function you write, (this includes any *auxiliary* functions you write) you must have a contract, purpose, and a minimum of 3 examples/test cases. The number of test cases required will depend on the task at hand.

1. Develop the function ***within?***, which consumes three numbers representing the x and y coordinates of a point and the radius of a circle centered at the origin. It returns true if the point is within or on the circle. It returns false otherwise. The distance of the point to the origin is the square root of $x^2 + y^2$.
2. Write a function, ***right-triangle?*** that takes 3 numbers as input. The three numbers represent the length of three sides of a triangle. The function should return true if the three sides can form a right triangle, false if not. The sides are not in any particular order.
3. Write a function, ***image-oriented***, which takes an image as input. It will output a string. If the image is taller than it is wide, the function outputs “**tall**”. If it is wider than it is tall, it will output “**wide**”. If the image width and height is the same, the function outputs “**square**”.
4. For this exercise, you will use the *2htdp/universe* and *2htdp/image* teachpacks. When developing your drawing function, use the ***place-image*** and ***empty-scene*** functions. It will make the integration of your functions with the animation function easier.

Create an animation of a circle (radius 10) that moves left to right across a 200x50 window. The circle should be vertically centered in the window. The circle changes color based on its position. In the first $\frac{1}{4}$ of the window, it is red, the second $\frac{1}{4}$ it is blue, the third $\frac{1}{4}$ it is orange and the last $\frac{1}{4}$ it is purple. Once it leaves the right side of the screen, it should wrap around and appear on the left again. To see what the animation should look like, click [here](#).

To accomplish this, create a functions ***render*** which draws your image in an empty-scene. The function will take a number for input. That input will represent the x position of the circle.

To implement your function, you will execute the following code:

```
(animate render)
```

DESSERT



5. A linear arithmetic progression is a series of numbers in which each successive number in the series can be formed by a linear expression from successive integers. For example, 100, 100.5, 101 forms a linear arithmetic progression, because the three numbers can be formed from the integers 1,2,3 by the expression $(0.5)x+99.5$. You can also determine if the ordered numbers form a progression by comparing the differences between successive numbers. Write a function, ***linear-progression?*** that takes as input 3 numbers. Given those three numbers, determine if they can form a linear arithmetic progression or not. Return true if they do form a progression, false if not. The numbers are ***not*** necessarily in any particular order, so you cannot assume they are in ascending or descending order. For example,
`(linear-progression? 5 2 8) should return true`
`(linear-progression? 2 4 5) should return false`
6. Define a function ***size*** that takes in a number, string, or image and returns how big it is. For a number, this means the absolute value of the number. For a string, this means the length of the string. For an image, it means the number of pixels (or area).
7. Define a function ***stretch-image*** that takes an image as input. If the image is taller than it is wide, the image should be stretched vertically by a factor of two. If it is wider than it is tall, it should be stretched horizontally by a factor of two. If its width and height are the same, the entire image should be stretched by a factor of two.
8. Create a new animation that works in a similar fashion to the circle animation from above. Instead of a circle moving across the screen, make the entire section of screen turn that particular color, in the same sequence – red, blue, green purple, but keeping the previous colors intact. Click [here](#) to see how the animation should work.