

COMPUTER SCIENCE

LAB 4

BOOLEANS AND CONDITIONALS

For each function you write, (this includes any *auxiliary* functions you write) you must have a contract, purpose, 2 examples, and a minimum of 3 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. A local discount store has a policy of putting labels with dates on all of its new merchandise. If an item has not sold within two weeks the store discounts the item by 25% for the third week, 50% for the fourth week, and 75% for the fifth week. After that no additional discounts are given.
Develop the function ***new-price***, which consumes the initial price of an item and the number of days since the item was dated and produces the selling price of the item.
3. Write a boolean 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.
4. For this exercise, you will use the *world.ss* teachpack, which incorporates the *image.ss* teachpack for drawing with animation functions. When developing your drawing functions, use the ***place-image*** and ***empty-scene*** functions from the teachpack. It will make the integration of your functions with the animation functions easier.

Develop a function called ***smart-dot*** that will draw a 200x200 pixel square. The square should be black with a inner white circle that has a radius of 75. The function will randomly draw a large dot (a circle with a radius of 10) somewhere within the confines of the 200x200 square. The function will consume two numbers. The numbers will correspond to the x and y coordinates of the center of the dot, *assuming the upper left corner of the square is the origin (0,0)*. The function will color the dot green if any part of it is within the white circle. The function will color the dot red if all of the dot lies outside the white circle (in the black).

Develop a function, ***random-smart-dot***, that will draw a smart-dot (as specified in the ***smart-dot*** function), but will generate random values for the x and y coordinates of the star. (hint: use the scheme random function to generate your random values). The entire dot must fit within the 200x200 square. The input will be a number. However, you

will not need to use the number in your function. (*It is necessary as input to run the simulation in the next step*).

Use the **run-simulation** function in the *world.ss* teachpack to run an animation of your **random-smart-dot** function. If you include the code:

```
(run-simulation 200 200 .5 random-smart-dot)
```

you should see an animation.

DESSERT



1. 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
```

2. Write a function, **evaluate-hand**, that will evaluate a poker hand. The input will be five values to represent five cards in a hand. The values may be a number value 2-10 or a symbol 'J, 'Q, 'K, or 'A to symbolize the face cards in a hand. Two assumptions are important to how the function works – first, no suits are considered so flushes are not possible outputs. Secondly, the cards will be input in order from low to high, so there is no need to sort the cards. The function will produce a symbol to represent the best hand derived from the cards, as follows (ranked from best to worst):

- 'four-of-a-kind – four cards with the same value
- 'full-house – three of a kind one value and a pair of another
- 'straight – five cards in sequential order
- 'three-of-a-kind – three cards with the same value
- 'two-pairs – two pairs with matching values
- 'pair – two cards of a single value
- 'nothing – meets none of the criteria above

Examples:

```
(evaluate-hand 4 7 7 7 'Q) would produce 'three-of-a-kind
```

`(evaluate-hand 8 9 10 'J 'Q)` would produce `'straight`

3. Make a new function, ***evaluate-hand-with-suits*** that is similar to the `evaluate-hand` function, but it takes 10 inputs that represent a five card hand. Every other input is a symbol representing the suit of the card ('S', 'H', 'C', 'D'). The output will be the same as those for ***evaluate-hand***, with the addition of suit consideration, as follows (ranked from best to worst):

- `'royal-flush` – ace, king, queen, jack, and 10, all of the same suit
- `'straight-flush` – a flush that is also a straight
- `'four-of-a-kind` – four cards with the same value
- `'full-house` – three of a kind one value and a pair of another
- `'flush` – all cards have the same suit
- `'straight` – five cards in sequential order
- `'three-of-a-kind` – three cards with the same value
- `'two-pairs` – two pairs with matching values
- `'pair` – two cards of a single value
- `'nothing` – meets none of the criteria above

Example:

`(evaluate-hand-with-suits 2 'H 3 'D 3 'H 'J 'S 'J 'H)`

would produce `'two-pairs`

because you have 2 of hearts, 3 of diamonds, 3 of hearts, Jack of spades, and Jack of hearts.