CS 131 - Homework 3

Deadline:

5:00 pm on Friday, September 17

How to submit:

When you are done with the following problems:

- save your resulting Definitions window contents in a file with the suffix **`.rkt`**
- transfer/save that file to a directory on **nrs-labs (**preferably in a folder/directory named `131hw3`)
- use **ssh** to connect to **nrs-labs**
- `cd` to the folder/directory where you saved it (`cd 131hw3` for example)
- use the `ls` command to make sure your `.rkt` file is really there
- use **`~st10/131submit`** to submit it, with a homework number of **3**
- make sure that `~st10/131submit` shows that it submitted your homework `.rkt` file

Purpose:

More practice writing functions using the design recipe; practice writing boolean functions/predicates and functions using a `cond` expression; practice writing functions that use auxiliary functions.

Important notes:

- Each student should work **individually** on this homework.
- You are expected to follow the Design Recipe for all functions that you write.
  - Remember, you will receive **significant** credit for the signature, purpose, header, and examples/check-expects portions of your functions.
  - Typically you'll get at least half-credit for a correct signature, purpose, header, and examples/check-expects, even if your function body is not correct
  - (and, you'll **lose at least half-credit** if you omit these or do them poorly, even if your function body is correct).
- Be especially careful to include at least one specific example/check-expect for each "kind"/category of data, and (when appropriate) for boundaries between data. You can lose credit for not doing so.

The Problems:

Problem 0

Start up DrRacket, setting the language to **Beginning Student** and adding the HTDP/2e versions of the `image` and `universe` teachpacks by putting these lines at the beginning of your Definitions window:

```
(require 2htdp/universe)
(require 2htdp/image)
```

Put a blank line, and then type in a comment-line containing your name, followed by a comment-line containing CS 131 - Homework 3, followed by a comment-line with no other text in it --- that is,

```
; type in YOUR name
; CS 131 - Homework 3
;
```

Problem 1

Below what you typed in Problem 0 above, type the comment lines:

```
; Problem 1
;
```

(Modified version of Exercise 4.2.1 from HtDP 1st edition, pp. 33-34)

A function that produces a value of type boolean is sometimes called a boolean function, and sometimes it is also called a predicate.

Using the design recipe, write a separate boolean function/predicate for each of the following intervals, that expects a number, and produces the boolean value `true` if the number is part of the interval described, and that produces the boolean value `false` if the number is not part of the interval described.

Note that:

- for this particular problem, the use of named constants for these particular intervals' end points is not required.
- a major part of this problem is developing enough specific examples/test cases (including all cases and boundaries between cases).
- hint: you should not need to use `cond` for these, although it is not incorrect if you do. The boolean operations `and`, `or` and `not` can be very useful here...

1 part a

Write the boolean function `is-within` to test if a number is within the interval (3, 7]

1 part b

Write the boolean function `in-either` to test if a number is in the union of (1, 3) and (9, 11). (For example, 2 is in this union; so is 9.5. But 3.7 is not...)

1 part c

Write the boolean function `is-outside` to test if a number is in the range of numbers **outside** of [1, 3].

Problem 2

Skip a line, and write a comment noting that what follows are your expressions for:

```
; Problem 2
```

```
;
```
A store gives discounts to frequent shoppers based on their past level of purchases; they are either "silver" level, "gold" level, or "platinum" level. Silver level frequent shoppers receive a 10% discount, gold level frequent shoppers receive a 25% discount, and platinum level frequent shoppers receive a 33% discount. All other shoppers receive no discount.

Using the design recipe, and remembering that the function `string=?` can be used to compare two strings for equality, design a function that expects a string representing the level of frequent shopper and produces the appropriate discount for that level written as a decimal fraction. (It should produce a discount of 0 if the string is not one of those noted above.) Be sure to include an appropriately complete set of specific examples/check-expects.

Problem 3

Skip a line, and write a comment noting that what follows are your expressions for:

```
; Problem 3
;
```
Using the design recipe, design a function that expects a string representing the level of frequent shopper and the total of a purchase before discount, and produces the appropriate discounted total for that purchase. For full credit, your solution must appropriately use your function from Problem 2.

Problem 4

Skip a line, and write a comment noting that what follows are your expressions for:

```
; Problem 4
;
```
(Adapted from Keith Cooper's section of Rice University's COMP 210, Spring 2002)

Conditionals (and Pizza Economics)

This problem considers an important consequence of increased pizza consumption –-the need for additional exercise.

Using the design recipe, develop a function `workout` that determines the number of hours of exercise required to counter the excess fat from eating pizza. `workout` expects a number that represents daily pizza consumption, in slices, and produces a number, in hours, that represents the amount of exercise time that you need.

For a daily intake of :	You need to work out for :
0 slices	1/2 hour
1 to 3 slices	1 hour
>3 slices	1 hour +1/2 hour per slice above 3

Be sure to include an appropriately complete set of specific examples/check-expects.

Problem 5

Skip a line, and write a comment noting that what follows are your expressions for:

```
; Problem 5
;
```
The remaining problems were adapted from a problem suggested by James Logan Mayfield on

the plt-edu mailing list.

Consider a virtual pet world: in this case, the number representing the world represents that pet's happiness.

Find or build an image to represent your virtual pet (you can create it from image operations, or paste one in, your choice). Define a named constant whose value is your virtual pet's image.

How can you show your pet's current happiness? You could convert the current number representing the world into a string, and then use the `text` operation to convert that string into an image of that text using a specified font size and color. Then you could use `above` to combine that text image and your pet image into a single image, representing your pet and its current happiness.

Now -- using the design recipe, design a `draw-pet-scene` function that expects a number, your pet's current happiness, and produces a scene containing your pet's image and a text image showing the value of your pet's current happiness.

Problem 6

Skip a line, and write a comment noting that what follows are your expressions for:

```
; Problem 6
;
```

You know that `big-bang` needs a function to give to its `on-tick` function, to say how the world should change at each clock tick. For our virtual pet's world, its happiness should decrease with each clock tick. Decide how much its happiness should decrease at each clock tick (your choice as to how much), and, using the design recipe, design a function `decr-happiness` that expects a number, your pet's current happiness, and produces the new decreased happiness that should result at the next clock tick.

Problem 7

Skip a line, and write a comment noting that what follows are your expressions for:

```
; Problem 7
;
```

What can make your poor virtual pet happy, then?

Feeding it makes it happy! Say that typing the key f -- represented as the string `"f"` -- should increase its happiness by 10.

Petting it makes it happy, too -- say that typing the key p -- represented as the string `"p"` -- should increase its happiness by 5.

But teasing it makes it less happy -- say the typing the key t -- represented as the string `"t"` -- should decrease its happiness by 3.

Typing any other key should result in the pet's happiness staying the same (not changing).

Using the design recipe, design a function `interact-w-pet` that expects the pet's current happiness and a string (representing a keystroke interaction), and produces the resulting new value of the pet's happiness that should result from that keystroke-interaction.

Problem 8

Skip a line, and write a comment noting that what follows are your expressions for:

```
; Problem 8
;
```

Now bring this all together by using the design recipe to design a function `main` that expects an initial pet happiness value, and starts up `big-bang` with at least:

- that initial pet happiness value,
- `decr-happiness` as its `on-tick` function (and whatever your desired choice of clock-speed will be)
- `draw-pet-scene` as its `on-draw` function
- `interact-w-pet` as its `on-key` function
- (and you may have additional `big-bang` clauses if you like)

This kind of `main` function is hard to write specific tests for -- so, no `check-expect`s are required for this `main` function.

Finally, call `main` at least twice, with at least two different initial pet happiness values.