Homework: Structures and variants

COMP 50

Fall 2013

Problems 1 to 4 of this homework are due at 11:59PM on **Monday, September 23**.
Submit them using the COMP 50 Handin button on DrRacket; the homework is the `structures` homework.

Problem 5 is due at 11:59PM on **Monday, September 30**. You will submit that problem either by signing a logbook located conveniently near the Earth's surface, or by submitting a `logbook` entry via the COMP 50 Handin button on DrRacket.

All the exercises should be done using the *Beginning Student Language*.

Finger Exercises

For this homework I am recommending the following finger exercises:

- In the First Edition *How to Design Programs*, Exercises 6.4.1, 6.4.3, 6.5.1, 6.6.1 (but instead of a symbol for color, please use a string), 6.6.3, 7.1.1, 7.1.3, 7.2.1, 7.2.2, 7.5.1, 7.5.3
- Develop data examples for the structure definitions in Exercise 6.4.1
- Write a function that converts a GPS position (as you define it below) to English

Problems to submit

1. Provide a *structure definition* and a *data definition* for representing clock time as a number of hours, minutes, and seconds elapsed since midnight. Unlike the example solution provided on the HtDP web site, your data definition must make clear exactly which values are and are not acceptable clock times.

   Define a function `move-big-hand` which adds one minute to a time structure.

   Define a function `time->digital` which consumes a time structure and produces an *image* showing how the time would read on a digital clock. Use the design recipe for a function that consumes compound data.

2. The Garmin eTrex handheld GPS renders distances in three ways:

   - Any distance up through 999 feet is rendered to the nearest whole foot, followed by the abbreviation "ft".
   - Any distance from 1000 feet up through 999 miles is rendered to the nearest *tenth* of a mile, using a decimal point and the abbreviation "mi".
   - Any distance of 1000 miles or more is rendered to the nearest mile, with no decimal point and the abbreviation "mi".

   Implement function `meters->english` which converts a distance in meters to a string showing the English units as on the Garmin GPS. Example outputs include `"37ft"`, `"850ft"`, `"0.2mi"`, `"10.0mi"`, and `"2268mi"`.

   Use the design recipe for conditional functions.

   Domain knowledge: 1 inch is 2.54 centimeters. 1 foot is 12 inches. 1 mile is 5280 feet.

3. Define a structure `gps-posn` and a data definition which represents GPS coordinates (latitude and longitude). Be sure that the data definition is sufficient so that anyone can determine whether an example structure is or is not a good GPS position.

   Define a function `checked-make-gps-posn` which behaves like `make-gps-posn` except that it guarantees that the latitude and longitude are good.

   Define a function `flat-earth-distance` that returns the distance in meters between two *nearby* GPS positions, assuming that near the positions you are interested in, the Earth's surface has been flattened. You may wish to consult the handout on the Earth's geometry.

   A *bearing* is a direction given relative to the compass. A bearing may be given as a string (N, S, E, W, NE, SE, NNW, and so on) or as a number of degrees. A bearing of zero degrees is North, ninety degrees is East, and so on. *Write a data definition for bearings*.

   Define a function which given two nearby GPS positions, returns the bearing of the second point with respect to the first. This bearing is the direction you are facing if you stand on the first point and look at the second. Again, assume that the Earth's surface has been pressed flat near the positions you are interested in.

   Define a function `gps-project` which consumes a position, bearing, and distance, and produces a new position obtained by starting at the first position, facing the given bearing, and traveling the given distance. This function should assume the Earth's surface has been flattened at the location of the initial point. This function (with a better model of the Earth) is built into my handheld GPS unit.

   Use both the design recipe for compound data and the design recipe for mixed data.

   Domain Knowledge: I've created a handout that shows the trigonometry needed to do the computations under the assumption that near the points of interest, the Earth's surface has been pressed flat.

4. While riding in his convertible with the top down, Professor Jacob is struck in the head by an errant baseball. When he recovers, he decides to hire you to create an experiment designed to confirm the validity of Fitts's Law, which you can look up on Wikipedia.

   Professor Jacob asks you to design a `big-bang` program that reacts to mouse clicks ("button-down" mouse events). In case the experimental subject decides not to click the mouse, the experimenter must also be able to limit the number of seconds for which the program runs.

   The program should draw the first mouse click as a seven-pointed, solid green star. At that point it should place a red circle at a random location on the canvas (display). The radius of the circle should be chosen randomly between 2 and 20 pixels.

   The program should then wait either for time to expire or for the experimental subject to click *within* the red circle.

   - If time expires, the program should write a consoling message over the canvas.
   - If the experimental subject successfully clicks within the red circle, the program should display a congratulatory message that includes the time that elapsed between the initial mouse click and the successful mouse click. Professor Jacob would really like to measure time to the nearest tenth of a second.

   After the message is written, the program should display its message for a few seconds, then return a structure containing the information relative to an experiment on Fitts's Law. Professor Jacob is still feeling a little foggy from the effects of the baseball, you will have to use the [Wikipedia page] to figure out what information to return and how to represent it.

   Use all of the design recipes above, plus a *wish list*.

   Domain knowledge: The Wikipedia page talks about a "device", a "target", and a "movement." The device is not under your control and you can't measure any of its properties directly. The "target" is the red circle you create and the starting point is the star. You are able to gather data about these aspects of the problem.

   Domain knowledge: You will need the `2hdtp/image` and `2htdp/universe` teachpacks. In the second edition, read Section 3.6 (Designing World Programs), especially Figure 10: Signatures for interaction functions.

- To understand how a program be limited to a given number of seconds, read the documentation for `on-tick`.
- To understand how to respond to the mouse, read the first part of Section 3.7 (A Note on Mice and Characters) in the second edition.

Special problem

5. **Tufts geography**: Later in the week I will create a teachpack that you can use with your `gps-project` function to follow directions over the part of the Earth's surface that contains Tufts. At the end of the directions you will find a box containing a log book where you can write your name. To help you in your travels, you will be permitted to use a phone, tablet, or other GPS device, as well as Google Earth or Google Maps.

To use the teachpack you will evaluate two expressions:

- Evaluating

  (`follow-waypoints-english` *make-gps-function latitude-selector longitude-selector passphrase*)

  will give you a rough set of directions in English.

- Evaluating

  (`follow-waypoints-computer gps-project` *make-gps-function latitude-selector longitude-selector passphrase*)

  will give you a GPS position computed using your `gps-project` function.

- If I can get all the pieces working,

  (`follow-waypoints-google-maps gps-project` *make-gps-function latitude-selector longitude-selector passphrase*)

  will take you to a Google Maps view of your destination.

Every student will have a different *passphrase* which I will cause to be emailed to you.

Traveling over the Earth's surface can be dangerous, especially if you are alone or it is dark outside. Therefore, the parts that involves finding the box and the log book are optional. The only required part of this problem is that you use the teachpack to produce GPS coordinates, which you then submit in human-readable form using *compass directions, degrees, minutes, and seconds.*

Karma problems

A. Define a function `time->analog` which converts a clock time into an image representing an analog clock. Your image must include an hours hand and a minutes hand; whatever other elements you wish to include are up to you.

B. Use `big-bang` to animate the passage of time on a clock. Run the animation at sixty times real time.

C. Run the Fitts's Law experiment multiple times in succession and use the results to estimate the characteristics of your device.

How your work will be evaluated

Here are the key questions:

1. Are there data descriptions where needed? Are they accompanied by data examples?
2. Does every function have an *appropriate* signature (called "contract" in the first edition), purpose statement, and header?
3. Did you come up with examples?
4. Are the signature and purpose statement precise enough that somebody else could come up with good examples?
5. Has the function body been developed by following a template that is suited to the problem and to the class of data being consumed?
6. In the coding step, have you carried out the purpose statement and only the one purpose statement?
7. Does your submission included test cases derived from your examples using `check-expect` and/or `check-within`? Are the test cases sufficient to handle every situation from your analysis and every clause in every relevant data description?