CS 2500, Lab 2

Online Submission

Refer to this page for a reminder on how to submit your labs and homework online. **Assignments will only be accepted electronically from this day forward!** Repeat this from your laptop or desktop at home if you wish to submit from it.

Pair Programming

In this lab, we will practice pair programming again. You will be working with your partner from last week. Like last time, each pair should only be working on one machine. Remember, in pair programming, one member of the team is the pilot and the other is the co-pilot. The pilot does the typing but the co-pilot drives the process. Even though the pilot is the only one typing, both partners should be active in trying to come up with solutions to the exercises. Make sure to switch roles when indicated in the lab!

Don't delete your work!
In previous labs some students deleted their solutions for exercises after completing themdon't do this! It is common for exercises to make use of functions or templates defined in earlier exercises.

Stepping through functions

Start DrRacket. Make sure that the language level is set to "Beginning Student" (*Note:* if you change the language level, it will not take effect until you hit the "Run" button).

Exercise 1: Design a function that consumes a natural number (0, 1, 2,), and returns a string representing it in unary (base 1). In unary, a number n is represented simply by repeating a symbol n times. Let's use the symbol "I". The number 2 would be represented by the string "II" and the number 5 would be "IIIII". To make these numbers more readable, we will also display the number in decimal after its unary form. Thus, 2 will be represented as "II (2)" and 5 as "IIIII (5)". The functions `replicate, number->string, and string-append` will be helpful here—look them up in the Help Desk.

Exercise 2: Design a function that consumes two natural numbers and produces a string illustrating how their unary representations add up (with the decimal form present as well). For instance, if given 2 and 4, it should produce the string "II (2) + IIII (4) = IIIIII (6)". Reuse the program you wrote for the previous exercise. If you don't, your code will be unreadable and take forever to write.

Exercise 3: The Stepper is a DrRacket tool that shows every step a program takes. Click on the Stepper button: the one labeled "Step" with a picture of a foot to the left of the Run button.

Use the Stepper on your programs from the last two exercises. Figure out what happens in the Stepper when one program calls another program.

Exercise 4: Run the Stepper on each of these programs:

```
(+ (sqr 2)
    (* (round 3.2)
       (- (sqrt 4)
          (/ 42 6))))
```

```
(place-image (circle 10 "solid" "black") 70 70
    (place-image (circle 10 "solid" "black") 30 70
       (place-image (rectangle 80 40 "solid" "red") 50 50
          (empty-scene 100 100))))
```

As far as DrRacket is concerned, what's the difference between an image and a number?

Switch Partner Roles

Pilots stand up and switch with your co-pilots.

Conditional Programs

Exercise 5: Design a function that calculates late fees on a movie rental. The function consumes the number of days the movie has been rented. Up to 3 days is a regular rental, no fee. For the next week the fee goes up $3 each day. Starting on the 10th day (i.e., a week late) there is a flat $20 fee that never changes.

Exercise 6: Run the Stepper on your program from the previous exercise. What happens in the Stepper when your program makes a conditional decision?

Switch Roles

Exercise 7: Design a function that calculates sales tax. The function consumes the sale price and the tax percentage (as a decimal or a fraction) and produces the final price. For instance, given 20 and 5/100 it should compute 21 (105% of $20).

Exercise 8: Design a function that calculates conditional sales tax: only prices $100 or more are taxed. The function consumes a sale price and a tax percentage and produces the final price. For instance, if given 20 and 5/100, it computes $20 (no tax). But if given 200 and 5/100, it computes $210 (5% tax). Hint: reuse your program from the previous exercise!

Exercise 9: Run the Stepper on your program that calculates conditional sales tax. Figure out where the Stepper calls your program that calculates unconditional sales tax. Try it with values both over and under $100.

Switch Roles

Playing with Posns

Posns are a kind of data provided by BSL that represent a position on a plane. As you would expect, a position on a plane is represented using two numbers, one for the value of the x-axis, and the other for the y-axis.

The `make-posn` function creates a posn. Use the help desk to find out how to use it. Once you have a posn, you will likely want to know what its x and y values are—to pass them to another function, for example. To do so, use the `posn-x` and `posn-y` functions, which give you the x and y values of the posn, respectively.

Add (require 2htdp/image) to the top of your file so we can use image functions like `circle` and `empty-scene`.

Exercise 10: Design a function `place-circle` that consumes a posn and produces a 300 300 scene with a red circle of radius 10 at the position represented by the posn. For instance, if given `(make-posn 10 290)` it should produce a scene with a red circle in the lower left corner.

Switch Roles

Mouse Clicks

In this part of the lab, we will create an interactive animation. Add `(require 2htdp/universe)` to the top of your file so we can use `big-bang` and friends. We've seen how to use the `to-draw`

and `on-tick` clauses before. Look them up in the Help Desk if you need to remind yourself how they work. Now we have one more function for reacting to the mouse pointer. Look up `on-mouse` in the Help Desk. Just like `to-draw` and `on-tick`, we have to give a function to `on-mouse`. Figure out what this function's inputs and outputs should be, and what a MouseEvent is.

Exercise 11: Design a function `mouse-click` that reacts to mouse clicks. It consumes a World, two numbers (x and y coordinates), and a MouseEvent, as described in `on-mouse`. In this lab, a World is a posn (make sure to note this in your program). When the MouseEvent is `''button-down''`, this function produces the x and y coordinates of the mouse click as a posn. Any other time, it produces the original World unchanged.

Exercise 12: Now that we have `mouse-click` and `place-circle`, let's create an animation. Add the following expression to your Definitions window, hit Run, and try clicking in the new window that opens.

```
(big-bang (make-posn 0 0)
    (to-draw place-circle)
    (on-mouse mouse-click))
```

If there is time left, try playing around with different MouseEvents to see what you can do with them. See if you can create an animation where...

- clicking the mouse creates a circle at the clicked position that expands.
- clicking down with the mouse creates a circle at the clicked position that expands until you stop pressing the mouse's button.
- clicking the mouse creates a circle that drops and falls off the scene.
- clicking the mouse creates a circle that drops and bounces off the bottom of the scene until it falls off the left or right sides of the scene.

Be creative, have fun with it!