Final Exam

This is the final exam for Intermediate Computer Science, Semester I, 2017-2018. Except as noted below, you may not use materials previously written by you or anyone else.

There are two sections: written (1/3) and programming (2/3). Before you use a computer, you must turn in your answers to the written questions.

Write all of your solutions on a separate sheet of paper.

Written Questions

(30 points: do only 3)

In the written portion, you may not consult any sources. Suggested time: 30 minutes.

1. (10 pts) Meg is writing a program. She is part way done, and now wants to finish the design and test it.

   ```
   (define (f x y any) (* x y))
   ```

   1. Write a signature.
   2. Write one good test for this function.
   3. Why might a function have a useless parameter like that?

2. Write a mouse handler that changes the amount of green in the model color whenever the x coordinate is greater than the y coordinate. In that case, the new amount of green is the difference in the mouse coordinates (`x-y`). Critique the mouse handler below. Suggest changes if needed for correctness.

   ```
   ; mouse-h: model x y -> number
   (define (mouse-h model x y)
     (cond [(> x y)
            (- x y)]
           [else
            (make-color (color-red model)
                        (color-green model)
                        (color-blue model))]))
   ```

3. (10 pts) Write a key handler that adds ten points to the model, leaving the rest alone.

   ```
   ;STRUCT moo: center = posn, points = number, clr = color
   (define-struct moo (center points clr))
   ```

   1. Signature
   2. One test.

3. Function.

4. (10 pts) Comment *in particular detail* about how the draw handler below will function when playing a kind of "click the dot" game. (What will you see? What will happen when you play the game?)

```
; purpose: place dot at a random position
(define (draw-h model)
  (place-image DOT
      (random 100)
      (+ 50 (random 200))
       BACKGROUND))
```

5. (10 pts) You are writing a game called Mondrian. The person playing places randomly colored squares on the screen.

   - As the mouse moves, the new colored square "floats" above the image so far.
   - They want to see every square placed since they start playing.

   1. Pick an appropriate structure to represent the model.
   2. Explain how the model allows you to write the draw-handler described above.
   3. Explain how the model allows you to write the mouse-handler described above.

Programming Questions

(60 points: do both)

In the programming portion, you may use the book *Picturing Programs*, the Racket Help Desk, your `posn-util.rkt` file, and the class blog.

Your work will be evaluated on the basis of correctness and how well it demonstrates your understanding of the design process.

Suggested time: 60 minutes.

1. **Random overlay**. Design and test a `random-overlay` function that takes in two images and then randomly places the first image on top of the second image. The placement will be done with `overlay/align`, choosing "left" "middle" "right"top" and "bottom" as appropriate. Make the middle location 50% likely to occur (both vertically and horizontally), and the other choices equally likely.

- Signature
- Purpose
- Check-expects covering all possibilities.

2. **Color match**. Show a colored rectangle and the numbers from an RGB code.

- Clicks reveals a green circle if they match, red circle if they do not match.
- Space bar (only) and gets a new randomly colored rectangle.
- This game would be impossible, but you need to "rig the game" so there about a 25% chance that they will match.
- It should be possible for every color and color code to appear.