CS 2500, Lab 4: Structures and Unions

New lab partners, new friends
It is time to switch lab partners! *You may not work with a partner with whom you have previously worked this semester*. Choose someone you haven't already worked with and relocate so that the two of you are sharing a single workstation.

Switching pilot and co-pilot
In previous labs you were switching pair-programming roles in between exercises. In this lab and in future labs, you will switch roles at regular intervals throughout the lab. The teaching assistants and tutors will announce that it is time to switch roughly every fifteen minutes.

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.

DR *uber alles*
Remember to practice using the design recipe when designing programs!

Develop means Design Recipe
Use it, love it.

Part I: Last week on CS 2500

Exercise 1: In this class, each lab section consists of a section number (1–6), a head TA, and a supporting TA. Design a data definition and define a data structure for representing lab sections. Give three examples of lab sections. (The course website has a table of all the lab sections and lists the names of all the TA's.)

Develop a program that consumes a lab section and returns a descriptive string similar to "CS2501 Section 2: Boboila, Simona".

Exercise 2: Develop a function that computes the result of the formula $(a+b)^2/(a-b)^2$. Give an example of using the function. For your example, (1) write down the evaluation steps in the *definitions window*, and then (2) use the stepper to compare how you evaluated the formula with how DrRacket evaluated the formula. Unless you have made a mistake, both ways of calculating the result of the formula should give the same vaule.

Part II: This week on CS 2500—Structures and unions

Hint: in the following exercises give names to your examples so you can use them again later.

Exercise 3: A rock band has a name and consists of a singer, a guitarist, a bassist, and a drummer. A jazz band has a name and consists of a trumpeter, a bassist, and a drummer. A pop band has a name and consists of a singer and a two synthesizer players. A band is either a rock band or a jazz band or a pop band. Write one data definition for each kind of band and then write a data definition for a band in general. Produce three examples of a band (one for each kind). Write a template for a function that consumes a band.

Exercise 4: A studio album has a name and a year of publication. A live album has a name, a year of recording, and a year of publication. A music album is either a live or a studio album. Write only one data definition to describe a music album. Produce two examples of a music album (one for each kind). Write a template for a function that consumes a music album.

Exercise 5: There are two kinds of unpublished music albums from the perspective of a music label—those that are completed and those that aren't. Both completed and uncompleted albums have a serial number, but those albums that are completed also have a name and a release date. Design a data definition to describe an unpublished music album. Produce two examples of an unpublished music album (one for each kind). Write a template for a function that consumes an unpublished music album.

Exercise 6: Develop a function that consumes an unpublished album and a serial number and checks whether the album matches the serial number.

Exercise 7: Develop a function that consumes an unpublished music album, a date of release, and a name. If the album is not completed it constructs an instance of a completed album using the information given as input to the function. If the album is already completed it returns it unchanged.

Part III: World with structures

In the following exercises, represent the world as a struct with two `posns`. The first `posn` represents the current position of a blue circle, and the second `posn` represents the current position of a red circle. When the user clicks the mouse the red circle will immediately move to where they clicked, and over time the blue circle will move to meet it.

Exercise 8: Design a function `mouse-handler` to react to mouse events. It consumes four inputs: a World, an x coordinate, a y coordinate, and a MouseEvent as described in MouseEvent. When the MouseEvent is "button-down" the function `mouse-handler` should create a World where the first `posn` is the same as the given World's first `posn` and the second `posn` is the position of the mouse click. On any other mouse event ("button-up", "drag", "move", "enter", or "leave") the function `mouse-handler` should return the given World unchanged.

Exercise 9: Design a function `tick-tock` to react to clock events. The purpose of the function is to gradually equate two `posns` as the clock progresses. The function consumes a World and produces a new World where both coordinates x and y of the first `posn` are increased or decreased by 1 (or 0) so that they approach the coordinates of the second `posn`.

For example, if the input World is ((1,3), (5,1)) then `tick-tock` should return the new World ((2,2), (5,1)).

Exercise 10: Design a function `world-draw` that consumes a World and returns a 300 300 scene with a solid blue circle of radius 15 at the position represented by the first `posn` and a solid red circle of radius 10 at the position represented by the second `posn`. When they overlap, the red circle should appear on top of the blue circle.

Exercise 11: Use `big-bang` and the three functions you wrote to create an animation where you click to place a red circle somewhere in the canvas and then a blue circle moves along the canvas trying to reach the red circle. The initial position of the blue circle is determined by how you choose to initialize the World. You'll need `on-mouse`, `on-tick`, and `to-draw`.

Exercise 12: Extend the definition of the world to include a String that is one of "red", "yellow", or "green". Modify `world-draw` to use this String to determine the color of the circle we draw when the user clicks. Also write a function `cycle-color` to cycle through the colors: "red" → "yellow" → "green" → "red". Using this function, modify `mouse-click` to cycle the color when the user clicks.

That's All Folks

If you had trouble finishing any of the exercises in the lab or homework, or just feel like you're struggling with any of the material, please feel free to come to office hours and talk to a TA or tutor for additional assistance.