CS 135 Winter 2019
Midterm Help Session
Reminder: Midterm (March 4)

- The midterm will be held on Monday, March 4 at 7:00 PM.
- Check your seating for the midterm on Odyssey.
- The midterm will cover up to and including the end of Module 06.
- There will be **NO** assignment due Tuesday, March 5.
Clicker Question - box-and-pointer

Which of the following nested box representations match this box-and-pointer representation?

A

B

C

D

E
Clicker Question - List Translation

Given this list:

(list (list) 'cons (list (list 2 'green) 3))

Which of the following is equivalent to the given list?

A  '(empty cons (list 2 'green) 3)
B  '(empty cons (2 'green) 3)
C  '(empty cons (list (list 2 'green) 3))
D  '(() cons ((2 green) 3))
E  '(() 'cons ((2 'green) 3))
Stepping through Lists

Give the first and second substitution steps as well as the final value for the following expression:

```
(length (rest (rest (second '((hello red) (0 1 1 2 3 5) () ()))))
```
Strings and Lists of Characters

Jimmy has decided to let Zahra change his legal name. However, since Jimmy forgot her birthday, she’s really angry and has decided to plot her revenge with the help of Dr. Racket. As she searches through a name (as a string) one character at a time, starting from the first character of the string, she modifies that string according to the following rules:

- Zahra has no issues with the empty string, and will not modify it in any way.
- Zahra loves seeing the letter Z appear in any string. Upon encountering a lower-case or upper-case Z character, the single occurrence of a Z will be replaced with two consecutive Z’s, both with the same case as the original Z.
• Zahra does not like the letter J, since it reminds her of Jimmy. As soon as Zahra sees a single occurrence of a lower-case or upper-case J, it is removed from the string. Zahra will stop searching through the rest of the string at that point, and simply produce the rest of the characters in the string (even if they should have been changed according to her rules).

• Finally, all other characters in the string will remain unchanged according to Zahra.

Write a function **Zahra-rename**, that consumes a string, and produces the result of modifying that string according to Zahra’s rules above. Note that it is appropriate to have a capital letter in the function name, as **Zahra** is a proper noun. You may find the function **char-ci=?** useful.
Zahra accidentally ordered a hundred storage boxes and has decided to organize her room. She wants to use a Racket structure to keep track of her stuff and which box it’s in. The box structure contains the following pieces of information:

- The type of items in the box,
- A list of items in the box (there must be at least one item),
- The number of items in the box, and
- The price of the box.

For example, `(define my-box (make-box 'Eye-Makeup (list "mascara" "eyeliner" "brow-gel") 3 21.75))` defines a box containing mascara, eyeliner, and brow-gel (all types of eye makeup).

Use `define-struct` to define the `box` structure. Provide a data definition as well.

How would Zahra pick out her eyeliner from `my-box`? Assume she knows that it is the second element in the list.
Template functions - Gosling

Consider the following definitions:

```
(define-struct canada-goose (height weight colour))
;; A Canada-Goose is a (make-canada-goose Num Num Sym)
;; requires: height, weight > 0

(define-struct grey-goose (height weight colour))
;; A Grey-Goose is a (make-grey-goose Num Num Sym)
;; requires: height, weight > 0
```

;; A Gosling is (anyof Canada-Goose Grey-Goose)

Write a template function for a Gosling and another template function for a list of Gosling.
Insertion Sort - sort-geese

Using your template functions as a guide, write a function called `sort-geese` that sorts a list of Geese in non-decreasing order of BMI. If two geese have the same BMI, they should appear in the same order as in the original list. The ordering of a canada-goose vs. a grey-goose does not matter. The BMI of a Gosling with height $h$ and weight $w$ is given by $w/h^2$. 
Recursing in lockstep - Destroy Recipes

Ben and Vincent are sad because Zahra changed Jimmy’s name to the empty string and they will avenge Jimmy by destroying all the recipes in her cookbook. A recipe in Zahra’s cookbook can be represented as a list of ingredients (which are symbols). They plan on merging two recipes with the same number of ingredients into one by writing a function, `destroy-recipe` that takes two recipes and produces a new recipe where all ingredients in an odd-numbered position are the ingredients that were in an odd-numbered position in the first recipe and all ingredients in an even-numbered position are ingredients that were in an even-numbered position in the second recipe. The ingredients in the produced recipe must be in the same relative order as they were in the original recipes. Remember that positions are numbered starting from zero!

```
(destroy-recipe '(flour sugar coconut sprinkles ice) '(eggplant lamb broccoli salt gravy))
⇒ '(eggplant sugar broccoli sprinkles gravy)
(destroy-recipe '(rice) '(milk)) ⇒ '(milk)
(destroy-recipe empty empty) ⇒ empty
```
Working with Lists

Write a function, `contains-between?` that takes in a list of numbers, a number, `value`, and two Nats, `low` and `high` and produces true if the number of occurrences of `value` in the given list is between `low` and `high` (inclusive), and false otherwise.
Zahra did not realize that Ben and Vincent destroyed her cookbook and held a disastrous dinner party for the CS 135 Staff. She is now going through her cookbook trying to remove all the destroyed recipes.

;; A Cookbook is one of:
;; empty
;; (cons (list Str (listof Sym)) Cookbook)

where the Str corresponds to the name of the recipe, and the listof Sym are the ingredients for a recipe. Zahra has the name of a recipe that she thinks has been destroyed. Write a function, remove-recipes, that removes the destroyed recipe from the Cookbook.

(\textbf{remove-recipes} \textit{"Pasta"} (\textbf{list} (\textbf{list} \textit{"Pasta"} \textit{'(ice chocolate milk)}))
\hspace{1cm} (\textbf{list} \textit{"Pizza"} \textit{'(crust sauce cheese)})))
\Rightarrow (\textbf{list} \textbf{list} \textit{"Pizza"} \textit{'(crust sauce cheese)}))

(\textbf{remove-recipes} \textit{"tuna"} (\textbf{list} \textbf{list} \textit{"Pickle"} \textit{'(())})) \Rightarrow (\textbf{list} \textbf{list} \textit{"Pickle"} \textit{'(())})
Mastering the Design Recipe

Following the design recipe helps you understand the problem and produce correct code. Here are the complete steps in tackling a problem using the design recipe:

- Read the question and summarize your task using a purpose.
- Next, determine the types for all of your input and output and express them in the form of a contract.
- Think of some valid input to the problem and calculate the output manually. These are your examples.
- After writing your function, test your function thoroughly by considering edge cases related to your function.
Templates

We write a template for functions that consume compound data. Here are a few things to pay attention to:

- If it is a structure, you need to select all of its fields.

;; songinfo-template: SongInfo → Any
(define (songinfo-template info)
  (... (songinfo-performer info) ...)
  (songinfo-title info) ... 
  (songinfo-genre info) ... 
  (songinfo-length info) ...))
Templates

- If it is a list, you need to consider whether it is an empty list. For a non-empty list, you need to process the first item and the rest of the list

```scheme
;; listof-str-template: (listof Str) → Any
(define (listof-str-template los)
  (cond [(empty? los) . . .]
        [else (. . . (first los) . . . (rest los) . . .)])
)
```

- In general, if your data definition has "one of..", which also includes lists, then include a conditional expression with one test for each possibility.
Recursion Strategies

In this course, we looked at the following ways of recursion:

- recursion on a number, such as adding numbers from 0 to n
- recursion on a list, such as adding numbers in a list
- recursion on a list and a number, such as getting the ith item of a list
- recursion on 2 lists (locked step or different rates), such as taking the dot product or merging 2 lists
- recursion on 2D-list/nested lists

To succeed in recursion questions, identify the type(s) of recursion you need to use and apply them effectively.
Optional - Recursing on a Nat3 - divisible-by-3?

Consider the following data definition for a natural number:

;; A Nat3 is one of
;; * 0
;; * 1
;; * 2
;; * (+ 3 Nat3)

Write a function called divisible-by-3? that consumes a Nat3 and produces whether the input number is divisible by 3. Use pure structural recursion.