Assignment Guidelines:

- For this and all subsequent assignments, you are expected to use the design recipe when writing functions from scratch, including helper functions.
- **For full marks, it is not sufficient to have a correct program. Be sure to follow all the steps of the design recipe. Read the Style Guide carefully to ensure that you are following the proper conventions. In addition, your solution must include the definition of constants and helper functions where appropriate.**
- Unless otherwise indicated in the question you may use only the built-in functions and special forms introduced in the lecture slides from CS115 up to and including the modules covered by this assignment. A list of functions described in each module of the lecture slides can be found at [https://www.student.cs.uwaterloo.ca/~cs115/built_in](https://www.student.cs.uwaterloo.ca/~cs115/built_in)
- Download the interface file from the course web page to ensure that all function names are spelled correctly, and each function has the correct number and order of parameters.
- Read each question carefully for restrictions.
- Test data for all questions will always meet the stated assumptions for consumed values.
- Do not copy the purpose directly from the assignment description. The purpose should be written in your own words and include references to the parameter names of your functions.
- The solutions you submit must be entirely your own work. Do not look up either full or partial solutions on the Internet or in printed sources.
- Do not send any code files by email to your instructors or tutors. Course staff will not accept it as an assignment submission. Course staff will not debug code emailed to them.
- You may post general assignment questions using the discussion groups on Waterloo LEARN. Choose Connect Discussions. Read the guidelines for posting questions. Do NOT post any code as part of your questions.
- Check Markus and your basic test results to ensure that your files were properly submitted. In most cases, solutions that do not pass the basic tests will not receive any correctness marks.
- Read the course web page for more information on assignment policies and how to organize and submit your work. Follow the instructions in the Style Guide.
- Your solutions should be placed in files a5qY.rkt, where Y is a value from 1 to 3.

**Plagiarism: The Following applies to all assignments in CS115:**

- Be sure to read the Plagiarism section at: [https://www.student.cs.uwaterloo.ca/~cs115/assignments](https://www.student.cs.uwaterloo.ca/~cs115/assignments)
Assignment 5
Due: October 24th, 2018
Language Level: Beginning Student
Coverage: Module 5

Note: Do not forget to add constraints for these questions as required to avoid common errors (division by zero etc.). Functions that violate constraints should return false, empty or 0 depending on the question.

Question 1:

Create a function `consecutive-elements?` that consumes `lst`, a list containing any type of values and returns true if two or more of the same value appear consecutively in the list, otherwise false.

Examples:

```lisp
(consecutive-elements? (cons 1 (cons "true" (cons (make-posn -1 3) (cons true (cons 5.6 empty)))))) => false
(consecutive-elements? (cons 1 (cons "true" (cons true (cons true empty))))) => true
```

Question 2

Create a function `sum-multiples-of-three-or` that consumes `loi`, a `(listof Int)` and `mult` a non-zero `Nat` and produces the sum of all numbers that divide evenly by 3, `mult`, or both.

Example:

```lisp
(sum-multiples-of-three-or (cons -21 (cons 15 (cons 12 (cons 3 (cons 14 (cons 2 (cons 8 empty))))))) 7) => 23
(sum-multiples-of-three-or empty 7) => 0
```

Question 3:

Create a function `filter-prefix` that consumes `los` a `(listof Str)` and `prfx`, a `Str` and produces a new `(listof Str)` with all words that begin with `prfx` removed.

This question should be approached in a case insensitive manner, meaning that “PrEflx” should be treated the same as “prefix.” You should create a helper function to ensure the correct answer. You are not allowed to use the built-in `string-downcase` or `string-upcase` or `string-ci=?` for this question. You may use built-in list functions, however, including `string->list`.

Example:

```lisp
(filter-prefix (cons "apple" (cons "b" (cons "banana" (cons "BAAnana" (cons "bAnAna" (cons "BaCkWaRdS" (cons "definite" empty))))))) "bA") =>
(cons "apple" (cons "b" (cons "definite" empty)))
```
(filter-prefix (cons "apple" (cons "b" empty)) "") => (cons "apple" (cons "b" empty))

**Question 4:**

Scientists sometimes “normalize” raw data to make all the values fall between two given values, while keeping the distances between the numbers proportional to each other. For example, a set of values \{0, 1, 2, 3, 4\} normalized between 0 and 100 will become \{0, 25, 50, 75, 100\}.

It is also common to transform abnormally large or small values in a data set (called outliers) to a pre-set maximum or minimum value. (Outliers are known to cause problems with certain statistics, such as calculating the mean.) So, \{-100000, 0, 1, 2, 3, 4\}, with mean -16663, restricted to values between 0 & 4 can be normalized to \{0, 0, 25, 50, 75, 100\} and therefore have a less obtrusive mean close to 42.

The formula for normalization you will use is

\[
\text{Normalized } x = 100 \times \frac{(x-\text{smallest})}{(\text{biggest}-\text{smallest})}
\]

Create a function `normalize-list raw biggest smallest` that consumes `raw`, a `(list of Int)`, and two `Ints`, `biggest` and `smallest`, and produces a new list containing values from 0 to 100, rounded to the nearest integer. Values larger than `biggest` should be treated as equal to `biggest`; values smaller than `smallest` should be treated as equal to `smallest` (to avoid outliers).

The normalization from 0 to 100 will be based on the range of values from `smallest` to `biggest`, regardless of the values that appear in the list. You should:

- Set the outliers to `biggest` or `smallest`.
- Subtract `smallest`.
- Divide by `(biggest - smallest)`
- Multiply by 100 and round.

Example:

(normalize-list (cons 55 (cons 5 (cons 50000 (cons 400 (cons 90 empty))))) 500 5) =>
(cons 10 (cons 0 (cons 100 (cons 80 (cons 17 empty))))))

(normalize-list (cons 55 (cons 5 (cons 90 empty))) 500 5) =>
(cons 10 (cons 0 (cons 17 empty))))