Assignment Guidelines.

- This assignment covers material in Modules 3 and 4.
- Submission details:
  - Solutions to these questions must be placed in files `a03q1.rkt`, `a03q2.rkt`, `a03q3.rkt`, and `a03q4.rkt`, respectively, and must be completed using Racket.
  - 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.
  - All solutions must be submitted to MarkUs. No solutions will be accepted through email, even if you are having issues with MarkUs.
  - Verify using MarkUs and your basic test results that your files were properly submitted and are readable on MarkUs.
  - For full style marks, your program must follow the CS115 Style Guide.
  - Be sure to review the Academic Integrity policy on the Assignments page.
  - Helper functions need design recipe elements but not examples and tests.
- Restrictions:
  - Unless the question specifically describes exceptions, you are restricted to using the functions and special forms covered in or before Modules 3 and 4.
  - Read each question carefully for additional restrictions.
- 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.
1. **Perfect Squares.**

Write a function \(\text{reduce-squares}\ \text{(L)}\) that consumes a \(\text{(listof Nat)}\). It returns a list containing the same values, except every number in \(\text{L}\) which is a perfect square is replaced with its square root.

\[
\text{(reduce-squares (list 2 3 4 5 6 7 8 9 10))} \Rightarrow \text{(list 2 3 2 5 6 7 8 3 10)}
\]

**Hint**

Read the documentation on the \text{integer?} predicate.

2. **Positives.**

Write a function \(\text{sum-positive}\ \text{(L)}\) that consumes a \(\text{(listof Int)}\) and returns the sum of all the positive numbers in the list.

\[
\text{(sum-positive (list 2 3 -5 4))} \Rightarrow 9
\]

3. **Broadcast Standards.**

It is desirable to remove all four-letter words from a body of text.

Write a function \(\text{censor}\ \text{(L)}\) that consumes a \(\text{(listof Str)}\). It returns a \text{Str} which contains all the words in \(\text{L}\), where each \text{Str} of length four has been replaced by "****", and spaces have been placed between the words.

\[
\text{(censor (list "When" "shall" "we" "three" "meet" "again;"}
     "In" "thunder," "lightning," "or" "in" "rain?")})
\Rightarrow "**** shall we three **** again; In thunder, lightning, or in rain?"
\]

\[
\text{(censor (list "When" "the" "hurlyburly's" "done,"}
     "When" "the" "battle's" "lost" "and" "won.")})
\Rightarrow "**** the hurlyburly's done, **** the battle's **** and ****"
\]

**Hint**

You may use the \text{join} function developed on a previous assignment: either your own code, or the code from the posted solution.

4. **The sort function.**

Sorting algorithms are often studied in Computer Science classes because the problem is easy to describe, and there is a wide range of interesting algorithms.

In real life, however, you should almost certainly use the built-in \text{sort} function.

The function \text{sort} consumes a list \(\text{L}\), and a predicate function \(\text{P}\). \(\text{P}\) consumes two arguments; if it returns \text{true}, the first argument will appear in the final list to the left of the second argument.

For example:

- \(\text{(sort (list 1 3 2 4)} <) \Rightarrow \text{(list 1 2 3 4)}\) since \((< 1 2), (< 2 3), \text{etc.}\)
- \(\text{(sort (list 1 3 2 4)} >) \Rightarrow \text{(list 4 3 2 1)}\) since \((> 4 3), (> 3 2), \text{etc.}\)
- \(\text{(sort (list -4 -3 -1 0 2 5)} (\lambda \text{a b}) (< \text{(abs a)} \text{(abs b)}))) \Rightarrow \text{(list 0 -1 2 -3 -4 5)}\)

Write a function \text{even-up-odd-down} that consumes a \(\text{(listof Int)}\), and returns the same items, sorted so it starts with even numbers, in increasing order, then has odd numbers, in decreasing order. Use the \text{sort} function to accomplish this.

\[
\text{(even-up-odd-down (list 11 7 1 3 10 6 5 2 8))} \Rightarrow \text{(list 2 6 8 10 11 7 5 3 1)}
\]