Assignment Guidelines.

- This assignment covers material in Module 5.

Submission details:
- Solutions to these questions must be placed in files a5q1.rkt, a5q2.rkt, and a5q3.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
- 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.
- For the design recipe, helper functions only require a purpose, a contract and an example.

Restrictions:
- Unless the question specifically describes exceptions, you are restricted to using the functions and special forms covered in or before Module 5.
- 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. Greatest Common Divisor. The Greatest Common Divisor (GCD) of two \( \text{Nat} \) is the largest \( \text{Nat} \) that both numbers are divisible by.

   (a) Listing all Divisors. Write a function \((\text{divisors-in \ n \ D})\) that consumes a \( \text{Nat} \) and a \((\text{listof \ Nat})\). The function returns a list containing all the values in \( D \) which \( n \) is divisible by.

   \[
   \text{(divisors-in \ 6 \ (list \ 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8))} \Rightarrow (\text{list \ 1 \ 2 \ 3 \ 6})
   \]

   (b) Common Divisors. Write a function \((\text{my-gcd \ a \ b})\) that returns the greatest common divisor of \( a \) and \( b \).

   \[
   \text{(my-gcd \ 42 \ 28)} \Rightarrow 14 \\
   \text{(my-gcd \ 0 \ 16)} \Rightarrow 16
   \]

   ! Do not use the built-in \( \text{gcd} \) or \( \text{lcm} \) functions.

2. Counting Pairs.

   (a) Pair item with list. Create a function \((\text{pair-item \ item \ L})\) that consumes an \( \text{Any} \) and a \((\text{listof \ Any})\), and returns a list containing all the \((\text{list \ Any \ Any})\), where the first value is \( \text{item} \), and the second value is taken from \( L \).

   For example,

   \[
   \text{(pair-item \ 4 \ (list \ 2 \ 3 \ 5))} \Rightarrow (\text{list \ (list \ 4 \ 2) \ (list \ 4 \ 3) \ (list \ 4 \ 5)}) \\
   \text{(pair-item \ "red" \ (list \ "cyan" \ "blue"))} \Rightarrow (\text{list \ (list \ "red" \ "cyan") \ (list \ "red" \ "blue"))}
   \]

   (b) All pairs. Create a function \((\text{make-pairs \ L1 \ L2})\) that consumes two \((\text{listof \ Any})\). The function returns a list containing all lists of two items, where the first item is taken from \( L1 \), and the second item is taken from \( L2 \).

   The lists should be in order, starting by pairing the first item in \( L1 \) with all items in \( L2 \), in order.

   \[
   \text{(make-pairs \ (list \ 1 \ 2) \ (list \ 3 \ 4))} \\
   \Rightarrow (\text{list \ (list \ 1 \ 3) \ (list \ 1 \ 4) \ (list \ 2 \ 3) \ (list \ 2 \ 4)})
   \]

   \[
   \text{(make-pairs \ (list \ 2 \ 7 \ 1 \ 8) \ (list \ "f" \ "u" \ "n"))} \\
   \Rightarrow (\text{list \ (list \ 2 \ "f") \ (list \ 2 \ "u") \ (list \ 2 \ "n") \ (list \ 7 \ "f") \ (list \ 7 \ "u") \ (list \ 7 \ "n") \ (list \ 1 \ "f") \ (list \ 1 \ "u") \ (list \ 1 \ "n") \ (list \ 8 \ "f") \ (list \ 8 \ "u") \ (list \ 8 \ "n"))
   \]

   Hint

   Use pair-item as a helper function, along with \( \lambda \).
3. **Sets.** A *set* is a collection of distinct objects. A *(listof Any)* may be used to represent a set; we simply require that there are no duplicates in the list.

    ;; a Set is (listof Any) which contains no duplicates.

For example, *(list 42 "Zaphod" 17 #false)* is a *Set*, while *(list 17 "Zaphod" 42 #false "Zaphod")* is not because "Zaphod" appears twice.

---

**Exercise**

(a) **Building.** Write a function *(make-distinct L)* that consumes a *(listof Any)*, and returns a *Set* containing the same items. The contract is:

    make-distinct: (listof Any) -> Set

Keep the copy of each item which is closest to the end.

For example,

    (make-distinct (list 17 "Zaphod" 42 #false "Zaphod")) => (list 17 42 #false "Zaphod")
    (make-distinct (list 2 4 6 8 7 6 4 2 4 5 6)) => (list 8 7 2 4 5 6)

Do not use *append* or *filter*.

**Hint**

Look up the documentation on the *member?* function.

---

(b) **Checking equality.** Write a function *(set=?)* that consumes two *Sets*, and returns #true if they contain the same items, not necessarily in the same order. For example,

    (set=? (list "Ford" 17 32 51) (list 32 51 "Ford" 17)) => #true
    (set=? (list 17 32) (list 32 42 17)) => #false
    (set=? (list 81 5 66) (list 66 5 81)) => #true