Assignment: 02
Due: Tuesday, September 29 at noon (Waterloo time)
Language level: Beginning Student
Files to submit: extremes.rkt, grades.rkt, pos.rkt, parity.rkt, mean-relative.rkt, sudoku.rkt
Practice exercises: HtDP 9.1.1, 9.1.2, and 9.1.3

- **Make sure you read the OFFICIAL A2 post on Piazza** for the answers to frequently asked questions.
- This assignment covers concepts up to the end of Module 06. Unless otherwise specified, you may only use Racket language features we have covered up to that point.
- Policies from Assignment 1 carry forward.
- For this and all subsequent assignments, you should include the design recipe as shown in the course content.
- The basic and correctness tests for all questions will always meet the stated assumptions for consumed values.
- You must use check-expect for both examples and tests of functions that produce exact values.
- **It is very important that your function names match ours.** You must use the basic tests to be sure. In most cases, solutions that do not pass the basic tests will not receive any correctness marks. The names of the functions must be written exactly. The names of the parameters are up to you, but should be meaningful. The order and meaning of the parameters are carefully specified in each problem.
- Since each file you submit may contain more than one function, it is very important that the code runs. If your code does not run, then none of the functions in that file can be tested for correctness.
- You may use examples from the problem description in your own solutions, however do not copy and paste them from the PDF as the formatting of the PDF may not be accepted by Dr. Racket.
1. **[10% Correctness]** In this question you will perform step-by-step evaluations of Racket programs, as you did in assignment one. Please review the instructions on stepping in A01. To begin, visit this web page:

   https://www.student.cs.uwaterloo.ca/~cs135/assign/stepping/

   **Note:** the use of `https` is important; that is, the system will not work if you omit the `s`. This link is also under the Assignments menu on the course web page.

   When you are ready, complete the two required questions under the **Module 4a: Boolean Operators** category, the three required questions under the **Module 4b: Conditionals**, and the three required questions under the **Module 6: Lists** category using the semantics given in the course content for Beginning Student.

2. For this question you will be writing several functions that operate on a non-empty list of numbers.

   (a) **[5% Correctness]** Write the function `smallest` that consumes a list of numbers and produces the smallest number in the list. Example:

   ```racket
   (smallest (cons -5 (cons 2 (cons 10.5 empty)))) => -5
   ```

   (b) **[5% Correctness]** Write the function `largest` that consumes a list of numbers and produces the largest number in the list. Example:

   ```racket
   (largest (cons -5 (cons 2 (cons 10.5 empty)))) => 10.5
   ```

   (c) **[5% Correctness]** Write the function `max-diff` that consumes a list of numbers and produces the largest difference between any two elements in the list. `max-diff` itself must not be called recursively. Instead you should make use of the previous two functions `smallest` and `largest`. Example:

   ```racket
   (max-diff (cons -5 (cons 2 (cons 10.5 empty)))) => 15.5
   ```

   Place your functions in `extremes.rkt`

3. **[10% Correctness]** Write the function `cs135-grade` that consumes five numbers `self-check, assign, mt1, mt2, and final` and produces the final grade received in CS135 out of 100 if those numbers are the student’s grades in decimal form for self-check exercises, assignments, midterm 1, midterm 2, and final assessment respectively.

   Recall that the formula for calculating your final grade in CS135 is available on the course webpage at https://www.student.cs.uwaterloo.ca/~cs135/assess/.

   Note that if the student does not pass the weighted exam or assignment portion of the class then they will receive either their normal grade or a 46 (whichever is lower), as they cannot pass the course without having passed these portions.

   Examples:

   ```racket
   (cs135-grade 0.8 0.6 0.4 0.9 0.95) => 68.3
   (cs135-grade 0.8 0.6 0.4 0.9 0) => 46
   ```

   Place your function in `grades.rkt`
A point-of-sale (POS) system is a software system used by a retailer to facilitate transactions between customers and the retailer. We’ll pretend we are working on a POS system in this question. For the next three questions about our POS system each function will consume two values. The first will be costs which will be a list of non-negative numbers that represents the individual prices of each item the customer wants to buy. The second input will be a non-negative number paid which represents the amount of money the customer is providing as payment.

Place all of your solutions in pos.rkt

(a) [5% Correctness] Write the function change-due that consumes the list of numbers costs and the non-negative number paid which represents the amount of money the customer is providing as payment.

change-due should produce the amount of change the customer should receive back (that is the sum of all of the costs deducted from the amount paid). You may assume for this question that customer will always provide enough money to cover the costs of the items they’re purchasing.

Examples:
(define prices1
  (cons 2.65 (cons 23.30 (cons 7.99 (cons 59.99 empty)))))
(change-due prices1 100) => 6.07
(change-due prices1 93.93) => 0

(b) [5% Correctness] Unlike above unfortunately sometimes people do not have enough money to cover the items they need to purchase. Write the function paid-enough? that consumes both the list of numbers costs and the non-negative number paid and produces whether or not the customer has paid enough to cover the costs. paid-enough? should produce true if paid is larger than or equal to the sum of costs, and false otherwise.

Examples:
(paid-enough? prices1 100) => true
(paid-enough? prices1 50) => false

(c) [5% Correctness] The POS system is for a local family-run store that believes in supporting and helping out those in their community. The owners of the store want functionality added to the POS system that allows them to easily find an item to give to the customer for free, which they can pay back at a time when they’re able to.

Write the function free-item that consumes the list of numbers costs and the non-negative number paid and produces the first item in the list that is large enough that if reduced to 0 would mean the customer has enough to pay for their bill. You may assume that there is always one item in the list large enough to cover the difference. You may also assume the customer has not provided enough to cover the costs.

Examples:
(free-item prices1 80) => 23.3
(free-item prices1 60) => 59.99
5. [10% Correctness] For this question we will consider what is called a “binary string” which is just a string of only 1’s or 0’s. In particular instances we can be concerned about the parity of a binary string, which simply means whether there is an even or odd number of 1’s in the string.

Write the function `parity` which consumes a string (which will only contain either the character `#\1` or `#\0`). Parity must return the symbol ‘odd if the number of 1’s in the string is odd, or the symbol ‘even if the number of 1’s in the string is even.

Examples:
```
(parity "110101") => 'even
(parity "1110011") => 'odd
```

The only string function you may use for this question is `string->list`.

Place your function in `parity.rkt`

6. [10% Correctness] A list of numbers might contain some that are below, equal to, or above the arithmetic mean of the numbers themselves.

Write the function `mean-relative` that consumes a list of integers and produces a list of symbols where each symbol in the produced list should be either ‘below-mean, ‘above-mean, or ‘mean if the number at location in the list was below the mean of the list, above the mean of the list, or equal to the mean of the list respectively.

Example: `(mean-relative (cons 5 (cons 7 (cons 9 (cons 12 empty))))))
```
=> (cons ‘below-mean (cons ‘below-mean (cons ‘above-mean
(cons ‘above-mean empty))))
```

Place your function in `mean-relative.rkt`

7. [10% Correctness] Sudoku is game where the player must enter the numbers one through nine into a 9x9 grid that is also broken up into 3x3 “boxes”. The goal of Sudoku is to make sure each row, grid, and box has the numbers one through nine in them (meaning there can be no duplicates as there are only nine spots in each container).

Write a function `sudoku-valid?` that consumes a list of numbers and produces true if the list of numbers contains only each of the numbers one through nine exactly once and nothing else, or produces false otherwise.

Examples:
```
(define valid (cons 1 (cons 2 (cons 3 (cons 4 (cons 5
(cons 9 (cons 8 (cons 7 (cons 6 empty))))))))))
(sudoku-valid? valid) => true
(sudoku-valid? (cons 1 valid)) => false
(sudoku-valid? (cons 0 valid)) => false
```

Place your function in `sudoku.rkt`
Challenges and Enhancements

Enhancement 1: For this enhancement you should extend your solution to Question 7. Write the function `spans-nat?` that consumes a list of numbers `lon` and a natural number `n`. `spans-nat?` should produce `true` if `lon` contains only each of the numbers one through `n` exactly once and nothing else, or produce `false` otherwise.

While the enhancement is just for your learning and you may solve it however you like, the challenge of this enhancement comes from using only concepts discussed up to Module 06. In particular, only use recursion that behaves the same as our simple structural recursion on a list (as opposed to recursing on a `Nat`, which is covered in Module 07). That means that in each recursive call you make the only change to you should make to an argument for the next call is to take the rest of the list.

For an additional challenge, do not use or implement a sorting algorithm to solve this question, since sorting is covered in Module 08.

Enhancement 2: You may already be familiar with the concepts of a mean and standard deviation from statistics. In question 6 you had to have calculated the mean of a list of numbers. The challenge here is to extend the work on statistics you’ve already done.

You should write a function `std-dev` that consumes a list of numbers and produces the standard deviation for that data set. The standard deviation can be calculated as a set of simple steps:

- Calculate the mean of the data set
- For each number calculate the square of the difference between the number and the mean
- Calculate the mean of all the numbers calculated in the second step
- Take the square root of the mean calculated in the third step

Once you have calculated the standard deviation for a data set you can start to talk about how far away any individual data point is from the mean, in terms of standard deviations. The z-score of a number measures how many standard deviations it is away from the mean of the data set it belongs to. The calculation for the z-score of a number is \( \frac{number - mean}{standard\ deviation} \). Write a function `z-scores` that consumes a list of numbers and produces a list of numbers which is the z-score for each number in the original list.