Winter 2025
Clarke

Assignment: 01

Due:	Tuesday, January 21, 2025 9:00 pm
Coverage:	End of Lecture 02
Language level:	Beginning Student
Files to submit:	`functions.rkt`, `jeopardy.rkt`, `conversion.rkt`

Assignment policies for this and all subsequent assignments

- The deadline for submission without penalty is stated above. More details are listed on the course website's assignment policies page.
- The work you submit must be entirely your own.
- Do not look up or generate either full or partial solutions on the Internet or in printed sources.
- Please comment your submissions as described in the lecture notes. Please read the course Web page for more information how to submit your work.

Grading

- **You will not receive marks for** *Ax* **(where** *x* **is 01, 02, ...) unless you have earned full marks for A00 before the** *Ax* **due date.** Assignments may be submitted as often as desired up to the due date.
- **Be sure to check your basic test results after each submission!** Your basic test results will be available shortly after you make a submission. Check it!
- If you do not get full marks on the basic tests, then your submission may not be markable by our automated tools, and you will very likely receive a low mark for the correctness portion of the corresponding assignment question.
- On the other hand, getting full marks on the basic tests does **not** guarantee your solution is correct. It only means that you spelled the name of the function correctly and passed some extremely trivial tests.

Here are the assignment questions you need to submit.

1. **(60%):** Functions

   Computers are helpful tools in many fields of human endeavour. Below are functions that are useful to compute, taken from a variety of areas.

   Implement the function definitions below in Racket, using the names given. Place your solutions in the file `functions.rkt`.

   Note that when you are asked to **implement** a function, it is best practice for it to be a direct implementation, though it can be any equivalent implementation. For example, when asked to implement $(a+b)$, the direct implementation is `(+ a b)`, but the implementation `(+ b a)` would be a valid equivalent implementation. When implementing x^2, the direct implementation is `(sqr x)`, but `(* x x)` would also be a valid equivalent implementation.

   For example, if we asked you to implement the function:

   $$mean(x1, x2) = \frac{x1 + x2}{2}$$

   you could submit:

   ```
   (define (mean x1 x2)
     (/ (+ x1 x2) 2))
   ```

   or:

   ```
   (define (mean x1 x2)
     (+ (/ x1 2)
        (/ x2 2)))
   ```

   But the one on the left is more direct and would be the better choice.

   (a) An example from geometry (*Manhattan distance*), write a function `manhattan-distance` that computes:

   $$\text{manhattan-distance}(x1, y1, x2, y2) = |x1 - x2| + |y1 - y2|$$

   (Note the hyphenated name *manhattan-distance*. Names in Racket can have hyphens in them for readability, unlike in some other computer languages, where `manhattan-distance` would be interpreted as `manhattan` minus `distance`.

   (b) An example from sports (*slugging average*):

   A baseball batter's slugging average is based on the number of singles s, doubles d, triples t, home runs hr and total at bats ab:

   $$\text{batter-slugging-average}(s, d, t, hr, ab) = \frac{s + 2 \times d + 3 \times t + 4 \times hr}{ab}$$

   Write a function `batter-slugging-average` that computes the above formula.

(c) Another example from geometry (*surface area of right circular cone*), write a function `cone-area` that computes:

$$\textit{cone-area}(r,h) = \pi r(r + \sqrt{h^2 + r^2})$$

Use the built-in value `pi`.

(d) An example from physics (*escape speed*):
The speed needed by a rocket to escape a planet's gravity is called the *escape speed*:

$$\text{escape}(M,r) = \sqrt{\frac{2GM}{r}}$$

where M is the mass of the body (e.g. planet) to escape and r is the body's radius. G is the gravitational constant $6.674 \times 10^{-11}\ \text{N} \cdot (\text{m}/\text{kg})^2$. This can be written in Racket as `6.674e-11`. Write a function `escape` that computes the above formula. As part of your solution, you should **`define`** G as a constant in Racket.

(e) An example from mathematics (*the partition function*):
A partition is a way of writing n as a sum of positive integers. For example $5+2+1+1$ is a partition of 9. Mathematicians Hardy and Ramanujan developed the following formula to approximate the number of partitions for the positive integer n:

$$\text{partition-size-approximation}(n) = \frac{1}{4n\sqrt{3}} \cdot e^{(\pi\sqrt{\frac{2n}{3}})}$$

Write a function `partition-size-approximation` that computes the above formula. Refer to the documentation for the difference between the Racket functions `exp` and `expt`.

(f) An example from finance (*Black-Scholes formula*), write a function `d1` that computes:

$$d1(\textit{maturity}, \textit{rate}, \textit{volatility}, \textit{spot-price}, \textit{strike-price}) =$$

$$\frac{1}{\textit{volatility} \cdot \sqrt{\textit{maturity}}} \cdot \left[\ln\left(\frac{\textit{spot-price}}{\textit{strike-price}}\right) + \left(\textit{rate} + \frac{\textit{volatility}^2}{2}\right) \cdot \textit{maturity} \right]$$

2. **(20%):** Speed

Using appropriate units for quantities such as distance, mass, or velocity is often important for real-world measurements. In fact, NASA's Mars Climate Orbiter crashed into Mars in 1999 because some of the programmers were assuming metric units while others were assuming imperial units![1]

In this question, you will write functions to convert between units. Place your solutions in the file `conversion.rkt`. Do not perform any "rounding". You do not have to worry about "divide by zero" errors.

[1] `https://llis.nasa.gov/llis_lib/pdf/1009464main1_0641-mr.pdf`, p.16

(a) The unit of speed most often used in physics is meters per second (m/s). A common imperial unit of speed is miles per hour (mph). Write a function `m/s->mph` that consumes a speed in the units of m/s and produces the same speed in units of mph. You must use the fact that one mile is exactly 1609.344 meters. Define a constant to help with the conversion. As a good test case, you should ensure that your function produces 3,600 (the number of seconds in an hour) when applied to 1609.344 (1 mile expressed in meters).

Reminder: Racket knows how to interpret simple fractions, so `1/3` is the same as `(/ 1 3)`. You can use this to your advantage when writing test cases.

(b) A more unusual unit of speed is *Smoots per millifortnight* (S/mfn). You must use the facts that one Smoot[2] is exactly $1.7018m$ and one millifortnight[3] is exactly $1209.6s$. Define a constant, or constants, to help with the conversion. Write a function `mph->s/mfn` that consumes a speed in units of mph and produces the same speed in units of S/mfn. As a good test case, you should ensure that your function produces $317\frac{1249}{1675}$ (i.e. $\frac{532224}{1675}$) when it is called with 1.

3. **(20%):** Jeopardy!

The game show *Jeopardy!* has three contestants who answer trivia questions. Each question has a monitary value. If the contestant answers the question correctly, the monitary value is added to their total. If they answer incorrectly, the value is subtracted from their total. The total starts at $0 at the beginning of the game.

At the end of the show there is a "Final Jeopardy" question. For this question each contestant must decide on a wager before the question is asked. An individual's wager can be any integer between 0 and their total (inclusive). This becomes the monitary value of the question for that contestant that is added or subtracted from their total.

For this question, we'll label the contestants as `c1`, `c2` and `c3`. We'll assume that all three have totals larger than $0 and that `c1`'s total is larger than the total of either `c2` or `c3`. The built-in Racket function `max` may be helpful.

(a) Write a function `min-wager` that consumes three numbers representing the total earned by `c1`, `c2` and `c3` (in that order). It produces the minimum that `c1` must wager to assure that he or she will have more money than either of the other two contestants, assuming that all answer the Final Jeopardy question correctly.

For example, `(min-wager 10000 7000 8000)` should produce `6001` because `c1` will then have $16,001 (assuming they answer correctly) while the most either of the other two contestants could have is $16,000.

[2] https://en.wikipedia.org/wiki/Smoot
[3] https://en.wikipedia.org/wiki/Fortnight

(b) Write a function `missed-question` that consumes three numbers as described for `min-wager`. The function produces the amount `c1` ends up with assuming (a) they use the `min-wager` strategy, and (b) they answer the Final Jeopardy question incorrectly.

For example, `(missed-question 10000 7000 8000)` should produce `3999`.

Place your solution in `jeopardy.rkt`.

This concludes the list of questions for you to submit solutions (but see the footnotes on the previous pages as well). Don't forget to always check the basic test results after making a submission.