MCS-177 Chapter 1 homework

Whenever you write any procedure for this class, the procedure should be tested throughly. For problems from Chapter 1, include those tests. For problems from other chapters, I will allow a redo when I find a bug only if you have shown your testing. In other words, if you have complete confidence that your testing was sufficiently thorough, you need not include your tests.

When a solution to a problem is a program, provide an explanation of the program if the logic is at all confusing. Generally, your programs should be easy to read without any explanation due to clear formatting and good choice of procedure and parameter names. When a solution is anything else, you should never simply provide an answer (such as a number); some explanation is required to justify your answer.

1a: (a) Write a Scheme expression **with no multidigit numbers** which has 3.14 as its value. (For the purposes of this problem, the number .02 has two digits.)

(b) Write a Scheme expression with no multidigit numbers which has 22/7 or $3\frac{1}{7}$ as its value.

1b: [HKK 1.4] According to the *Joy of Cooking*, candy syrups should be cooked 1 degree cooler than listed in the recipe for each 500 feet of elevation above sea level.

(a) Define `candy-temperature` to be a procedure that takes two arguments: the recipe's temperature in degrees and the elevation in feet. It should calculate the temperature to use at that elevation. The recipe for Chocolate Caramels calls for a temperature of 244 degrees; suppose you wanted to make them in Denver, the "mile high city." (One mile equals 5280 feet.) Use your procedure to find the temperature for making the syrup.

(b) Candy thermometers are usually calibrated only in integer degrees, so it would be handy if the `candy-temperature` procedure would give an answer rounded to the nearest degree. Rounding can be done using the predefined procedure called `round`. For example, `(round 7/3)` and `(round 5/3)` both evaluate to 2. Insert an application of `round` at the appropriate place in your procedure definition and test it again.

1c: The built in Scheme procedures `max` and `min` compute the maximum and minimum of their arguments. For example, `(max 5 2)` returns 5. Write a succinct English description of what the following procedures return. Try to express *what* each calculates, not *how* it calculates that.

(a)
```
(define puzzle1
  (lambda (a b)
    (- (+ a b)
       (max a b))))
```

(b)
```
(define puzzle2
  (lambda (a b c)
    (min (min (max a b)
              (max b c))
         (max a c))))
```

To help you see the difference between *what* and *how*, for Exercise 1.7(b), and unsatisfactory answer would be, "The procedure first checks if x is less than 0, and if it is it subtracts x from 0, otherwise it just returns x." The better answer is, "The procedure returns the absolute value of x."

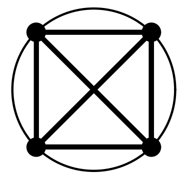
1d: [HKK 1.16] A 10-foot-long ladder leans against a wall, with its base 6 feet away from the bottom of the wall. How high on the wall does it reach? This question can be answered by evaluating `(ladder-height 10 6)` after entering the following definition. Make a diagram such as the one in Figure 1.1 showing the evaluation of `(ladder-height 10 6)` in the context of this definition:

```
(define ladder-height
  (lambda (ladder-length base-distance)
    (sqrt (- (square ladder-length)
             (square base-distance)))))
```

1e: If n points are placed on a circle, there are $\binom{n}{2}$ ways of choosing a pair of points to connect, where

$$\binom{n}{2} = \frac{n(n-1)}{2}.$$

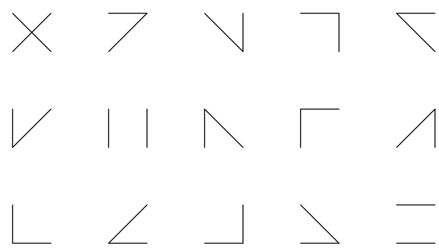
For example, if $n = 4$, $\binom{n}{2} = \frac{4(4-1)}{2} = 6$. The 6 lines are shown below:



The number of possible pairs of lines would be $\binom{\binom{n}{2}}{2}$. For $n = 4$,

$$\binom{\binom{n}{2}}{2} = \binom{6}{2} = 15.$$

The possible 15 pairs of lines are:



(a) Write a procedure `n-choose-2` that takes one argument, n, and computes $\binom{n}{2}$. Test your procedure on at least 2 inputs.

(b) Write a procedure `n-choose-2-choose-2` to compute $\binom{\binom{n}{2}}{2}$ using the procedure `choose`. Keep your procedure short and simple by calling the procedure `n-choose-2` which you wrote for part (a). Test your procedure on at least 2 inputs.

Note that in problems such as this one, *testing* a procedure does not mean the same as *running* a procedure. You should confirm that the procedure returns the correct answers! You could do this by hand-calculating answers *or* (not as good) asking a neighbor if her procedure gave the same answers.