Homework 1 Due Feb 2nd 11:59 pm

You should submit a single .scm file with all definitions on learn

All answers which are not Scheme definitions should be written using comments (;;)

All exercises are from the textbook, The Scheme Programming Language (https://www.scheme.com/tspl3/)

- Exercise 2.2.1
- Using the symbols *a* and *b* and the function *cons* we could construct the list (*a b*) by evaluation the following expression: (*cons* 'a (*cons* 'b '())). Using the symbols *a*, *b*, *c*, *d*, and the functions *cons*, construct the following lists **without** using quoted lists(i.e. '(*a b*)).
  - (*a b c d*)
  - (*a* (*b c d*))
  - (*a* (*b c*) *d*)
  - ((*a b*) (*c d*))
  - (((*a*)))
- Exercise 2.2.3
- Exercise 2.3.1
- Exercise 2.4.1
- Exercise 2.5.1
- Exercise 2.5.3
- Exercise 2.6.2
- Exercise 2.6.3 (only implement caar, cdar, caaar, caadr)
- What is the value of the following expressions?
  - (*and #t* (*or #t #f*))
  - (*or #f* (*and* (*not #f*) *#t #t*))
  - (*not* (*or* (*not #t*) (*not #t*)))
  - (*and* (*or #t #f*) (*not* (*or #t #f*)))
- Exercise 2.7.1
- Exercise 2.7.2
- Exercise 2.8.3
- Exercise 2.8.4 (name your functions my-list-ref and my-list-tail)
- Define a function *subst-first* that takes the following three arguments: an item *new*, an item *old*, and a list of items *ls*. The function should look for the first top-level (do not recurse into sublists) occurrence of the item *old* in the list and replace it with *new*. For example:
  - (subst-first 'dogs 'cats '(I love cats)) => (I love dogs)
  - (subst-first 'x 'y '(+ x y)) => (+ x x)
  - (subst-first 'x 'y '(+ x (* y y) y)) => (+ x (* y y) x)
  - (subst-first '(hello) '(world) '(hello world (world))) => (hello world (hello))
  - (subst-first 'a 'b '()) => ()
- Define a function *firsts* that takes a list of nonempty lists. This function

should return a list of the first item from each sub-list. For example:
  - (firsts '((a) (b) (c))) => (a b c)
  - (firsts '((a b c) (d e f) (g h i))) => (a d g)
  - (firsts '(((uno)))) => ((uno))
  - (firsts '()) => ()
- Define a function *remove-last* that removes the last top-level occurrence of a given *item* in a list *ls*. For example:
  - (remove-last 'i '(m i s s i s s i p p i)) => (m i s s i s s i p p)
  - (remove-last 'i '(m i s s i s s i p p i s)) => (m i s s i s s i p p s)
  - (remove-last 'i '()) => ()