LIGN 165
Problem Set 1

Please put your answers into a file called `lastname-firstname.clj`. Be careful to follow the instructions exactly and be sure that all of your function definitions use the precise names, number of inputs and input types, and output types as requested in each question. **If you do not follow these instructions, we cannot give you credit for your answer.**

For running your code, you have two options. You can run your code on the course website. Alternatively, you can install Clojure on your local machine and write and debug the answers to each problem in a local copy of `lastname-firstname.clj`. You can find information about installing and using Clojure here `https://clojure.org/`.

Problem 1: Write a procedure called `abs` that takes in a number, and computes the absolute value of the number. It should do this by finding the square root of the square of the argument.

Note: you should use the `Math/sqrt` procedure built in to Clojure, which returns the square root of a number. For example, to get the square root of 4, you can call:

```
(Math/sqrt 4)
```

Problem 2: In both of the following procedure definitions, there are one or more errors of some kind. Explain what's wrong and why, and fix it:

```
(defn take-square
  (* x x))

(defn sum-of-squares [(take-square x) (take-square y)]
  (+ (take-square x) (take-square y)))
```

Problem 3: The expression `(+ 11 2)` has the value `13`. Write four other different Clojure expressions whose values are also the number `13`. Using `def` name these expressions `exp-13-1`, `exp-13-2`, `exp-13-3`, and `exp-13-4`.

Problem 4: Write a procedure, called `third`, that selects the third element of a list. For example, given the list `'(4 5 6)`, third should return the number `6`.

Problem 5: Write a procedure, called `compose`, that takes two one-place functions `f` and `g` as arguments. It should return a new function, the composition of its input functions, which computes `f(g(x))` when passed the argument `x`. For example, the function `Math/sqrt` (built in to Clojure from Java) takes the square root of a number, and the function `Math/abs` (also built in to Clojure) takes the absolute value of a number. If we make these functions Clojure native functions using `fn`, then `((compose Math/sqrt Math/abs) -36)` should return `6`, because the square root of the absolute value of `-36` equals `6`.

```
(defn sqrt [x] (Math/sqrt x))
(defn abs [x] (Math/abs x))
((compose sqrt abs) -36)
```

Problem 6: Write a procedure `first-two` that takes a list as its argument, returning a two element list containing the first two elements of the argument. For example, given the list `'(4 5 6)`, `first-two` should return `'(4 5)`.

Problem 7: Write a procedure `remove-second` that takes a list, and returns the same list with the second value removed. For example, given `(list 3 1 4)`, `remove-second` should return `(list 3 4)`

Problem 8: Write a procedure `add-to-end` that takes in two arguments: a list `l` and a value `x`. It should return a new list which is the same as `l`, except that it has `x` as its final element. For example, `(add-to-end (list 5 6 4) 0)` should return `(list 5 6 4 0)`.

Problem 9: Write a procedure, called `reverse`, that takes in a list, and returns the reverse of the list. For example, if it takes in `'(a b c)`, it will output `'(c b a)`.

Problem 10: Write a procedure, called `count-to-1`, that takes a positive integer `n`, and returns a list of the integers counting down from `n` to `1`. For example, given input `3`, it will return `(list 3 2 1)`.

Problem 11: Write a procedure, called `count-to-n`, that takes a positive integer `n`, and returns a list of the integers from `1` to `n`. For example, given input `3`, it will return `(list 1 2 3)`. Hint: Use the procedures `reverse` and `count-to-1` that you wrote in the previous problems.

Problem 12: Write a procedure, called `get-max`, that takes a list of numbers, and returns the maximum value.

Problem 13: Write a procedure, called `greater-than-five?`, that takes a list of numbers, and replaces each number with `true` if the number is greater than `5`, and `false` otherwise. For example, given input `(list 5 4 7)`, it will return `(list false false true)`. Hint: Use the function `map` that we discussed in class.

Problem 14: Write a procedure, called `concat-three`, that takes three sequences (represented as lists), `x`, `y`, and `z`, and returns the concatenation of the three sequences. For example, given the sequences `(list 'a 'b)`, `(list 'b 'c)`, and `(list 'd 'e)`, the procedure should return `(list 'a 'b 'b 'c 'd 'e)`.