# LING/COMP 445, LING 645
# Problem Set 1

**Name**: , **McGill ID**:
*Collaborators*:

Due before 4:35 PM on Wednesday, September 13, 2023

Please enter your name and McGill ID above. There are several types of questions below.

- For questions involving answers in English or mathematics or a combination of the two, put your answers to the question in the answer box provided, like in the example below.

  This .pdf document you are reading was compiled from a .tex document with LaTeX. You must use LaTeX for this problem set, and future problem sets in this course.[1]

- For programming questions, please put your answers into a file called `ps1-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.

  For the code portion of the assignment, **it is crucial to submit a standalone file that runs**. Before you submit `ps1-lastname-firstname.clj`, make sure that your code executes correctly without any errors when run at the command line by typing `clojure ps1-lastname-firtname.clj` at a terminal prompt. We cannot grade any code that does not run correctly as a standalone file, and if the preceding command produces an error, the code portion of the assignment will receive a 0.

  To do the computational problems, we recommend that you install Clojure on your local machine and write and debug the answers to each problem in a local copy of `ps1-lastname-firstname.clj`. You can find information about installing and using Clojure here `https://clojure.org/`.

Once you have entered your answers, please compile your copy of this LaTeX file into a PDF and submit

(i) the compiled PDF renamed to `ps1-lastname-firstname.pdf`
(ii) the raw LaTeX file renamed to `ps1-lastname-firstname.tex` and
(iii) your `ps1-lastname-firstname.clj`

to the Problem Set 1 folder under 'Assignments' on MyCourses.

---

**Example Problem:** This is an example question using some fake math like this $L = \sum_0^\infty \mathcal{G}\delta_x$.

**Note:** If you're new to LaTeX, find the code corresponding to this text in the .tex document to see an example of using LaTeX to typeset math (you'll need this for later problem sets).

**Example Answer:** Put your answer in the box provided, like this:

| Example answer is $L = \sum_0^\infty \mathcal{G}\delta_x$. |
|---|

[1] To compile a file `file.tex` to `file.pdf`, you can install LaTeX, and use the command `pdflatex file.tex` at the command line, or make use of an online service such as `https://overleaf.com`. You can find more information about LaTeX here `https://www.latex-project.org/`.

**Problem 1:** Write an expression which defines a variable `year` with the integer value `2023`.

**Answer 1:** Please put your answer in `ps1-lastname-firstname.clj`.

**Problem 2:** Consider the following:

```
(= 4 (+ 1 2))
```

This is:

A. an expression
B. a list
C. both
D. neither

**Answer 2:** Please put your answer in the box below.

**Problem 3:** Which of the following evaluates to a value (A, B, both, or neither)?

A. `'(2 2 2)`
B. `(2 2 2)`

**Answer 3:** Please put your answer in the box below.

**Problem 4:** Consider the following expression:

```
(= "4" (+ 1 3))
```

This expression contains:

A. a string
B. quoted material
C. both
D. neither

**Answer 4:** Please put your answer in the box below.

**Problem 5:** Write a function `add-up` that takes two numbers returns their sum.

**Answer 5:** Please put your answer in `ps1-lastname-firstname.clj`.

**Problem 6:** Write a function `is-it-four?` that returns `true` when given the number `4`, and returns `false` otherwise.

**Note**: Don't forget the question mark in the function name! This is a convention in Clojure[2] for the names of predicate functions (functions that return a boolean value—`true` or `false`). Also, an incorrectly named function won't be seen by the grader script, so even if your function behaves correctly, you won't get credit if it has the wrong name!

**Answer 6:** Please put your answer in `ps1-lastname-firstname.clj`.

---

**Problem 7:** Fill in the blank, so the following expression evaluates to `true`:

```
(= (quote ___) 'platypus)
```

**Answer 7:** Please put your answer in `ps1-lastname-firstname.clj`.

---

**Problem 8:** Define a function `func` and an expression `expr` such that the following evaluates to `true`.

```
(= 3 (apply func expr))
```

**Hint:** be sure you understand what kinds of arguments `apply` expects.

**Answer 8:** Please put your answer in `ps1-lastname-firstname.clj`.

---

**Problem 9:** The built-in function `type` is useful for checking what kind of object an expression evaluates to.[3] Write a function `both-same-type?` that takes two arguments, and returns `true` when they both have the same type, and `false` otherwise.

**Answer 9:** Please put your answer in `ps1-lastname-firstname.clj`.

---

**Problem 10:** Write a function `list-longer-than?` which takes two arguments: an integer `n`, and a list `lst` and returns `true` if `lst` has more than `n` elements, and `false` otherwise. For example, `(list-longer-than? 3 '(1 2 3))` should return false, and `(list-longer-than? 2 '(1 2 3))` should return true.

**Hint:** you may find built-in clojure function `count` useful.

**Answer 10:** Please put your answer in `ps1-lastname-firstname.clj`.

---

[2] See `https://guide.clojure.style/#naming-predicates`.

[3] Though note that the types are different in Clojure (in the command line) versus ClojureScript (in the textbook). For instance, they disagree about whether integer numbers (like 4) and floating point numbers (like 4.0) are different types. **For the purposes of this problem set, don't worry about this**.

**Problem 11:** In linear algebra, if $\mathbf{x}, \mathbf{y}$ are two vectors each with $n$ components, their dot product is $\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i y_i$. Write a function `dot-product` that takes two lists of numbers as arguments, and returns the dot product. So for example, if the list `x` is `'(0 2 4)` and the list `y` is `'(1 3 5)`, the expression `(dot-product x y)` should return `26`, because $0 \cdot 1 + 2 \cdot 3 + 4 \cdot 5 = 26$.

You may assume that the two input lists are of equal length and contain only numbers as elements.

**Hint:** you may find built-in clojure functions `apply` and `map` useful.

**Answer 11:** Please put your answer in `ps1-lastname-firstname.clj`.

---

**Problem 12:** In Clojure (like other functional programming languages) functions and variables are treated identically. This means a function may easily take another function as an argument, and/or return a function.[4] Write a function `swap-arg-order` which takes a *function* (of two arguments) as an argument returns *another function* that does the same thing, but expects its two arguments in the opposite order.

That is, for example

- given the division function `/` which divides the first argument by the second (so `(/ 3 6)` returns the number `1/2`), the following expression should evaluate to `2`

  ```
  ((swap-arg-order /) 3 6)
  ```

- given the function `list-longer-than?` from above, the following expression should evaluate to `true`

  ```
  ((swap-arg-order list-longer-than?) '(1 2 3) 2)
  ```

**Answer 12:** Please put your answer in `ps1-lastname-firstname.clj`.

---

**Problem 13:** Define a higher order function `g` so the following expression evaluates to `true`:

```
(= 100 (g (fn [n] (* n n))))
```

**Answer 13:** Please put your answer in `ps1-lastname-firstname.clj`.

---

[4] Functions which take functions as arguments or return functions are called ‘higher order functions’. Built-in functions `map` and `apply` are higher order functions.