LING/COMP 445, LING 645
Problem Set 2

Name: Mathieu-Joseph Magri , **McGill ID**: 260928498

Due before 8:35 AM on Thursday, February 2, 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 an answer box like in the example below.
- For programming questions, please put your answers into a file called `ps2-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 `ps2-lastname-firstname.clj`, make sure that your code executes correctly without any errors when run at the command line by typing `clojure ps2-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 `ps2-lastname-firstname.clj`. You can find information about installing and using Clojure here https://clojure.org/.

  Note there is a built-in function called `reverse`. **Do not use the built-in function `reverse` in this problem set!** We remove `reverse` from the namespace when we grade, so using it anywhere will lead to an error and a result in a 0.

Once you have entered your answers, please compile your copy of this LaTeX file[1] into a PDF and submit

(i) the compiled PDF renamed to `ps2-lastname-firstname.pdf`
(ii) the raw LaTeX file renamed to `ps2-lastname-firstname.tex` and
(iii) your `ps2-lastname-firstname.clj`

to the Problem Set 2 folder under 'Assignments' on MyCourses.

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

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 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 a single-argument function called `absval` that, when passed a number, computes its absolute value. It should do this by finding the square root of the square of the argument. (Note: you should use the `Math/sqrt` function built in to Clojure (from Java), which returns the square root of a number.)

Answer 1: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 2: In both of the following definitions, there are one or more errors of some kind. In each case, 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)))
```

Answer 2: Please put the fixed functions in `ps2-lastname-firstname.clj` and **describe what is wrong and why in the box below**.

For "take-square", when using "defn" instead of "fn", we must still define our inputs using "[]". In the original function, we do not not initialize the inputs and therefore the function is not declared properly. By adding "[x]", the function is fixed and works as intended.

For "sum-of-squares", when using the original function, we get a "Unsupported binding form" error at line [(take-square x) (take-square y)]. We instead should just simply pass x and y by doing [x y]. By doing so, the function is fixed and works as intended, by suming the squares of both input numbers.

Problem 3: The expression `(+ 11 2)` evaluates to `13`. Write four other different Clojure expressions which also evaluate to the number `13` (either the integer `13` or the float `13.0`). Using `def`, assign these expressions to the symbols `exp-13-1`, `exp-13-2`, `exp-13-3`, and `exp-13-4`.

In each def statement, be sure to quote the expression (as below for our example), so it is not evaluated before being assigned to the symbol.

```
(def exp-13-0 '(+ 11 2))
```

Answer 3: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 4: Define a function called `third`, that selects the third element of a list. For example, given the list `'(4 5 6)` as its argument, `third` should return the number `6`.

Answer 4: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 5: Define a function 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` of `g` of `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` (likewise) takes the absolute value of a number. If we use `defn` to define functions `sqrt` and `abs` as

```
(defn sqrt [x] (Math/sqrt x))
(defn abs [x] (Math/abs x))
```

then `((compose sqrt abs) -36)` should return `6`, since the square root of the absolute value of `-36` equals `6`.

Answer 5: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 6: Define a function called `first-two` that takes a list as its sole argument, and returns a two-element list containing the first two elements of the argument. For example, given the list `'(4 5 6)`, `first-two` should return the list `'(4 5)`.

You may assume that the list passed in has at least two elements.

Answer 6: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 7: Define a function called `remove-second` that takes a list, and returns a new list that is the same as the input list, but with the second value removed. For example, given `'(3 1 4)`, `remove-second` should return the list `'(3 4)`.

Answer 7: Please put your answer in `ps2-lastname-firstname.clj`.

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

Answer 8: Please put your answer in `ps2-lastname-firstname.clj`.

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

Do not use the built-in function `reverse` (in this problem, nor anywhere in this problem set).

Answer 9: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 10: Define a function 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 the list `(list 3 2 1)`.

Answer 10: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 11: Define a function 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 the value of `(list 1 2 3)`.

Hint: Use the procedures `reverse-list` and `count-to-1` that you wrote in the previous problems.

Answer 11: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 12: Define a function called `get-max`, that takes a list of numbers, and returns the maximum value. So, `(get-max '(2 3 3))` should return `3`.

Don't use the built-in `max` function.

Answer 12: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 13: Define a function called `greater-than-five?`, that takes a list of numbers, and returns a list of equal length to the input list, but where each number is replaced with `true` if the number is greater than `5`, and `false` otherwise. For example, given input `(list 5 4 7)`, it will return the list `'(false false true)`.

Hint: Use the built in function `map`.

Answer 13: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 14: Define a function 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 arguments `(list 'a 'b)`, `(list 'b 'c)`, and `(list 'd 'e)`, the procedure should return the value of `(list 'a 'b 'b 'c 'd 'e)`.

Don't use the built-in `concat` function.

Answer 14: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 15: Define a function called `sequence-to-power`, that takes a sequence (represented as a list) `x`, and a nonnegative integer `n`, and returns the sequence $\mathtt{x}^n$. For example, given the sequence `(list 'a 'b)` as the first argument and the number `3` as the second, the procedure should return the value of `(list 'a 'b 'a 'b 'a 'b)`.

You may use the built-in `concat` function for this problem.

Answer 15: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 16: Define L as a language containing a single sequence, $L = \{a\}$.

In Clojure, we can represent the sequence a as the list `'(a)`.

Define a function called `in-L-star?` that takes a sequence (represented as a list), and returns true if and only if the sequence is a member of the language L^*. For example, given a sequence such as `'(a b)`, the procedure should return `false`, because ab is not a member of L^*.

Answer 16: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 17: Let A and B be two distinct formal languages. We'll use $(A \cdot B)$ to denote the concatenation of A and B, in that order. Find an example of languages A and B such that $(A \cdot B) = (B \cdot A)$.

Answer 17: Please put your answer in the box below.

If we define A as the formal language containing the empty string $A = \{\epsilon\}$, and $B = \{a\}$, then $(A \cdot B) = (B \cdot A)$ given that the empty string ϵ concatenated with any other string gives back the same element, since ϵ is the identity element of concatenation much like 1 is the identity element of multiplications in mathematics.

Problem 18: Let A and B be languages. Find an example of languages A and B such that $(A \cdot B)$ does not equal $(B \cdot A)$

Answer 18: Please put your answer in the box below.

If we define A as $A = \{a\}$, and B as $B = \{b\}$, then $(A \cdot B) \neq (B \cdot A)$ given that $(A \cdot B) = \{ab\}$ while $(B \cdot A) = \{ba\}$.

Problem 19: Find an example of a language L such that $L = L^2$, i.e. $L = (L \cdot L)$.

Answer 19: Please put your answer in the box below.

If L is defined as $L = \{\epsilon\}$, then $L = L^2$, since $L^2 = L \cdot L = \{\epsilon^2\} = \{\epsilon \cdot \epsilon\} = \{\epsilon\}$.

Problem 20: Argue that the intersection of any two languages L and L' is always contained in L.

Answer 20: Please put your answer in the box below.

An intersection, by definition, is a set operation that creates a set of elements between two sets for which all elements in the created intersection set are present in both sets. That is when doing $A \cap B$ for two sets A and B, we create a set that contains all elements of A that are also in B. Therefore, by definition, $A \cap B$ is a subset of A. The same logic can be applied to a language L. That is, if we do $L \cap L'$, we get the set that contains all elements of L that are also in L', and thus, by definition, $L \cap L'$ is a subset of L and is therefore contained by L.

Example: We have two sets $A = \{1, 2, 3\}$ and $B = \{2, 3, 4\}$. $A \cap B = \{2, 3\}$. As we can see, all elements of $A \cap B$ are contained within the set A.

Problem 21: Let L_1, L_2, L_3, and L_4 be languages. Argue that the union of Cartesian products $(L_1 \times L_3) \cup (L_2 \times L_4)$ is always contained in the Cartesian product of unions $(L_1 \cup L_2) \times (L_3 \cup L_4)$.

Answer 21: Please put your answer in the box below.

When doing the union of languages L_1 and L_2, we effectively combine all elements in both sets in to one set. The same can be said for when we do the union of languages L_3 and L_4. Hence, when doing the Cartesian product $(L_1 \cup L_2) \times (L_3 \cup L_4)$, we effectively go through all elements in the set L_1 and create pairs with all elements in both L_3 and L_4, and we do the same with L_2, meaning we go through all elements in L_2 and create pairs with all elements in both L_3 and L_4. In other words, $(L_1 \cup L_2) \times (L_3 \cup L_4)$ is equivalent to $(L_1 \times L_3) \cup (L_1 \times L_4) \cup (L_2 \times L_3) \cup (L_2 \times L_4)$. Therefore, we can

clearly see that $(L_1 \times L_3) \cup (L_2 \times L_4)$ is contained in $(L_1 \cup L_2) \times (L_3 \cup L_4)$, given that $(L_1 \times L_3) \cup (L_2 \times L_4)$ is the set that contains all possible pairs between the sets L_1 and L_3 and all possible pairs between the sets L_2 and L_4.

Problem 22: Let L and L' be finite languages. Show that the number of elements in the Cartesian product $L \times L'$ is always equal to the number of elements in $L' \times L$.

Answer 22: Please put your answer in the box below.

When doing a cartesian product between two sets, every element of one set is associated to one element of the other set. These associations are done for every single element in each set. Therefore, if the set L_1 has a cardinality of 5 and the set L_2 has a cardinality of 3, $L_1 \times L_2$ will have a cardinality of 15 given that all 5 elements of L_1 will have made 3 pairs each given that L_2 has a total of 3 elements. When doing $L_2 \times L_1$, although we don't get the same elements as when doing $L_1 \times L_2$ given that order of operations has an influence on the resulting set when doing a Cartesian product, the cardinality of $L_2 \times L_1$ will be equal to the cardinality of $L_1 \times L_2$, since all three elements of L_2 will have made 5 pairs each with the 5 elements of L_1. Therefore, for two sets L and L', $|L \times L'| = |L' \times L|$.

Note: The cardinality of a Cartesian product is simply the product of the cardinalities of each individual set. Therefore, from the above explanation, $|L_1 \times L_2| = |L_1| \cdot |L_2|$. (" $\cdot$ " here symbolizes the regular, mathematical multiplication operator)

Problem 23: Suppose L is a language, and that concatenation of L with itself is equal to itself: $(L \cdot L) = L$. Show that L is either the empty set, the set $\{\epsilon\}$, or an infinite language.

Answer 23: Please put your answer in the box below.

We have three situations.

If $L = \varnothing$, then we know that the empty set concatenated with itself (and concatenated to all elements for that matter) returns the empty set, thus itself. Therefore, $L \cdot L = \varnothing^2 = \varnothing = L$

If $L = \{\epsilon\}$, that is the set containing the empty string, we know that the empty string concatenated to any element including itself returns the element that it was concatenated with. Therefore, $L \cdot L = \{\epsilon^2\} = \{\epsilon\} = L$

Finally, if we take an infinite set such as $L = V^*$, the Kleene star set on V, then we know that $L \cdot L = (V^* \cdot V^*) = V^* = L$, given that when multiplying two infinite things together, we always end up with an infinity of things.