LING/COMP 445, LING 645
Problem Set 2

Due before 10:05 AM on Thursday, September 30, 2020

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** section 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/.

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 right under the question like this $L = \sum_0^\infty \mathcal{G}\delta_x$.

---

[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 here**.

For the first procedure, there is no parameter vector.

For the second procedure, a function is applied on the parameters before they are bound to values giving a syntax error. Even if that were to work, it would compute the sum of powers of four, not of squares.

Problem 3: The expression `(+ 11 2)` has the 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` as well as here.

```
(def exp-13-1 '(- 15 2))
(def exp-13-2 '(* 3.25 4))
(def exp-13-3 '(/ 26 2))
(def exp-13-4 '(+ 7 6))
```

Problem 4: Write a procedure, 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: 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` 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 make these functions Clojure native functions using `defn`,

```
(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: Write a procedure `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: Write a procedure `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: Write a procedure `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: Write a procedure, called `reverse`, 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)`.

Note there is a built-in function called `reverse`. Do not use this function in this problem set! Also note (without alarm) that the REPL will warn you[2] that you are replacing a built-in function.

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

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 the list `(list 3 2 1)`.

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

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 the value of `(list 1 2 3)`.

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

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

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

[2] something like `WARNING: reverse already refers to: 'clojure.core/reverse ... being replaced by: 'user/reverse`.

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

Problem 13: Write a procedure, 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: 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 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)`.

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

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

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)`.

Write a procedure `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 here.

$$\begin{gathered}\text{Let } A = \{a, b\} \text{ and } B = \{\epsilon\} \\ \Rightarrow (A \cdot B) = \{a \cdot \epsilon, b \cdot \epsilon\} \text{ and } (B \cdot A) = \{\epsilon \cdot a, \epsilon \cdot b\} \\ \therefore (A \cdot B) = (B \cdot A) = \{a, b\}\end{gathered}$$

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 here.

$$\text{Let } A = \{a, b\} \text{ and } B = \{c\}$$
$$\Rightarrow (A \cdot B) = \{a \cdot c, b \cdot c\} \text{ and } (B \cdot A) = \{c \cdot a, c \cdot b\}$$
$$\therefore (A \cdot B) \neq (B \cdot A)$$

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 here.

$$\text{Let } L = \{\epsilon\}$$
$$\Rightarrow L^2 = \{\epsilon \cdot \epsilon\} = \{\epsilon\}$$
$$\therefore L = L^2$$

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

Answer 20: Please put your answer here.

$$\text{Let } I = L \cap L' = \{x | x \in L \wedge x \in L'\}$$
$$\Rightarrow \forall x \in I, x \in L$$
$$\therefore \text{The intersection of any two languages L and L' is always contained in L}$$

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 here.

By definition,

$$(L_1 \times L_3) \cup (L_2 \times L_4) = \{a \cdot b | (a \in L_1 \wedge b \in L_3) \vee (a \in L_2 \wedge b \in L_4)\}$$
$$(L_1 \cup L_2) \times (L_3 \cup L_4) = \{a \cdot b | (a \in L_1 \vee a \in L_2) \wedge (b \in L_3 \vee b \in L_4)\}$$

It suffices to argue that the logical conditions on objects belonging to $(L_1 \cup L_2) \times (L_3 \cup L_4)$ are met for all possible members of $(L_1 \times L_3) \cup (L_2 \times L_4)$

Case 1:

$$\text{Suppose } a \in L_1 \wedge b \in L_3$$
$$\Rightarrow (a \in L_1 \vee a \in L_2) \wedge (b \in L_3 \vee b \in L_4)$$

Case 2:

$$\text{Suppose } a \in L_2 \wedge b \in L_4$$
$$\Rightarrow (a \in L_1 \vee a \in L_2) \wedge (b \in L_3 \vee b \in L_4)$$

Case 3:

$$\text{Suppose now that } a \in L_1 \wedge a \in L_2 \wedge b \in L_3 \wedge b \in L_4$$
$$\Rightarrow (a \in L_1 \vee a \in L_2) \wedge (b \in L_3 \vee b \in L_4)$$

$$\therefore (L_1 \cup L_2) \times (L_3 \cup L_4) \subseteq (L_1 \times L_3) \cup (L_2 \times L_4) \text{ given case 1, 2, 3}$$

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 here.

$$\text{Suppose } |L| = n \text{ and } |L'| = m \text{ for some } m, n \in \mathbb{N}$$
$$\Rightarrow |L \times L'| = n \cdot m \text{ and } |L' \times L| = m \cdot n$$
$$\therefore |L \times L'| = |L' \times L|$$

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 here.

We can show the above by examining each possible case. Namely, $L = \{\}$, $L = \{\epsilon\}$, $|L| = \infty$ or any arbitrary L that is not one of the three aforementioned cases.

Case 1: $L = \{\}$

$$\text{It is trivial to see that } (L \cdot L) = \{\} \text{ since this amounts to concatenating nothing with nothing}$$
$$\therefore (L \cdot L) = L$$

Case 2: $L = \{\epsilon\}$

$$(L \cdot L) = \{\epsilon \cdot \epsilon\} = \{\epsilon\}$$
$$\therefore (L \cdot L) = L$$

Case 3: $|L| = \infty$

$$\text{Let } L \text{ be a language defined over some vocabulary } V \text{ and } |L| = \infty$$
$$\text{We know } L \subseteq V^* \Rightarrow L = V^* \text{ since } |L| = |V^*|$$
$$\text{Now, suppose } (L \cdot L) \neq L$$
$$\Rightarrow \exists a \in (L \cdot L) \text{ s.t } a \notin L$$
$$\text{But this cannot be since } L = V^* \text{ and } V^* \text{ is the set of all possible strings over vocabulary } V$$
$$\therefore (L \cdot L) = L$$

Case 4: L is any other language

$$\text{Let } L \text{ be any arbitrary finite language where } |L| > 0 \text{ and } L \neq \{\epsilon\}$$
$$\text{Suppose that } (L \cdot L) = L$$
$$\text{Consider } L' = \{a, b\}$$
$$\Rightarrow (L' \cdot L') = \{aa, ab, ba, bb\}$$
$$\therefore (L' \cdot L') \neq L' \rightarrow\leftarrow$$
$$\therefore (L \cdot L) \neq L \text{ for some arbitrary finite language where } |L| > 0 \text{ and } L \neq \{\epsilon\}$$

$\therefore$ If $(L \cdot L) = L,$ then L must be either the empty set, the set $\{\epsilon\}$ or an infinite language by case 1, 2, 3, 4