LING/COMP 445, LING 645
Problem Set 1

Jessica Chan

There are several types of questions below. For programming questions, 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.

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 `lastname-firstname.clj`. You can find information about installing and using Clojure here `https://clojure.org/`.

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. You can find more information about LaTeX here `https://www.latex-project.org/`.

Once you have answered the question, please compile your copy of this LaTeX into a PDF and submit (i) the compiled PDF (ii) the raw LaTeX file and (iii) your `lastname-firstname.clj` via email to *both* timothy.odonnell@mcgill.ca and savanna.willerton@mail.mcgill.ca. Please make sure all of your code compiles and runs, we cannot give partial credit for code that won't run at all.

Problem 0: This is an example question using some fake math like this $L = \sum_0^\infty \mathcal{G}\delta_x$.

Answer 0: $L = \sum_0^\infty \mathcal{G}\delta_x$.

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.)

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

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)))
```

Answer 2: The function abs is missing the argument `[x]` that should be passed. The function sum-of-squares' arguments should not have the take-squared function called on them. They should look like this instead: `[x 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`.

```
(def exp-13-1 []
      (* 13 1))
(def exp-13-2 []
      (+ 3 10))
(def exp-13-3 []
      (+ (expt 3 2) 4))
(def exp-13-4 []
      (- 15 2))
```

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`.

Answer 4: Please put your answer in `firstname-lastname.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(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)
```

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

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)`.

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

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)`

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

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)`.

Answer 8: Please put your answer in firstname-lastname.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 `'(a b c)`, it will output `'(c b a)`.

Answer 9: Please put your answer in firstname-lastname.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 `(list 3 2 1)`.

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

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

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

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.

Answer 13: Please put your answer in firstname-lastname.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 sequences `(list 'a 'b)`, `(list 'b 'c)`, and `(list 'd 'e)`, the procedure should return `(list 'a 'b 'b 'c 'd 'e)`.

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

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

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

Problem 16: Define L as a language containing a single sequence, $L = a$. Write a procedure `in-L?` that takes a sequence (represented as a list), and decides if it is a member of the language L^*. That is, given a sequence x, the procedure should return `true` if and only if x is a member of L^*, and `false` otherwise.

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

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

Answer 17: Let $A = a, aa$ and $B = aaaa, aaa$. Then $AB = aaaaa, aaaa, aaaaaa$. Which is equal to $BA = aaaaa, aaaa, aaaaaa$.

Problem 18: Let A and B be languages. Find an example of languages A and B such that $\textsc{concat}(A, B)$ does not equal $\textsc{concat}(B, A)$

Answer 18: Let $A = a, ab$ and $B = b, ba$. Then $AB = ab, aba, abb, abba$. Which is not equal to $BA = ba, bab, baa, baab$.

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

Answer 19: Let $L = a, ab, aba$.
$L^2 = aa, aab, aaba, abab, ababa, abaaba$ and
$LL = aa, aab, aaba, abab, ababa, abaaba$. We can see the two are equal.

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

Answer 20: The intersection of languages L and L' is a subset equal to $L \cap L'$. Breaking this down, we know that any element $x \in L \cap L'$ must belong in both sets by definition of an intersection. So, all elements $x \in L \cap L'$ will also be 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 $(L_1 \cup L_2) \times (L_3 \cup L_4)$. $L' \times L$

Answer 21: We know that any elements in the arbitrary pair $(x, y) \in (L_1 \times L_3) \cup (L_2 \times L_4)$ are either a Cartesian product of $(L_1 \times L_3), (L_2 \times L_4)$ or both by the definition of union.

From $(L_1 \cup L_2) \times (L_3 \cup L_4)$ we can deduce that any pair $(m, n) \in (L_1 \cup L_2) \times (L_3 \cup L_4)$ will be a cross product of $m \in (L_1 \cup L_2)$ and $n \in (L_3 \cup L_4)$.

So, we can see that the possible cross products of this union are $(m, n) \in (L_1 \times L_3) \cup (L_2 \times L_3) \cup (L_1 \times L_4) \cup (L_2 \times L_4)$ by the distributive property of union.

We can see that if $(x, y) \in (L_1 \times L_3) \cup (L_2 \times L_4)$, then $(x, y) \in (L_1 \times L_3) \cup (L_2 \times L_3) \cup (L_1 \times L_4) \cup (L_2 \times L_4)$ also since the former is a subset of the latter. Thus the Union of Cartesian products is always in a Cartesian product of Unions.

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: We know that the number of elements in some Cartesian product of $(A \times B)$ is given by $|A| \bullet |B|$. So, by the commutative property of multiplication, we know that the number of elements in $L \times L'$ is equal to the number of elements in $L' \times L$, since $|L| \bullet |L'| = |L'| \bullet |L|$

Problem 23: Suppose that the concatenation of a language L is equal to itself: $\texttt{concat}(L, L) = L$. Show that L is either the empty set or an infinite language. (Remember that the empty set contains the null string.)

Answer 23: Let L be the empty set. We know that concatenating the empty set to the empty set will yield the empty set since sets are unique.

Now, let I be the infinite set ab^{∞}. Concatenating this infinite set to itself will yield more of the same pattern ab, however any "new" iterations of this repetition will already be a part of the original infinite set. Since sets are unique and any "new patterns" concatenated are mere pointers to the original instantiation, nothing new is added. Therefore, concatenation of I to I yields itself.