

**CS3342 – Assignment 4**  
**due Apr. 7, 2022**  
**2-day no-penalty extension until: Apr. 9, 11:55pm**  
**(SRA's cannot be used to extend the due date further)**

1. (15pt) Consider the following Prolog rule:

`a(X, Y) :- b(Z), c(X, Z), d(X, Y, Z), e(Z, W).`

Rewrite this rule in predicate calculus, using:

- (a) four universal quantifiers;
  - (b) two universal quantifiers and two existential quantifiers;
  - (c) four existential quantifiers.
2. (25pt) Implement a sorting algorithm (of your choice) in Prolog as a predicate `my_sort(X, Sorted_X)`, working as shown below:

```
?- my_sort([1], X).  
X = [1] .  
?- my_sort([2,1,3], X).  
X = [1, 2, 3] .
```

Submit your code as a file `my_sort.pl`.

3. (60pt) Implement the following Scheme functions:

```
(define my-permutations  
  (lambda (L) ...  
(define list-lex-less?  
  (lambda (L1 L2) ...  
(define my-sort  
  (lambda (L comp-pred?) ...
```

`my-permutations` generates a list of all permutations of a given list (in any order); `list-lex-less?` compares two lists and outputs true iff the first is smaller than the second in lexicographic order; `my-sort` sorts a given list (with an algorithm of your choice) using the given `comp-pred?` predicate to compare elements. The three functions are then put together into `sorted-permutations` to produce all permutations of a given list in lexicographical order.

```
(define sorted-permutations  
  (lambda (L) ...
```

Here are some examples:

```
(my-permutations '(1 3 2)) => '((1 3 2) (3 1 2) (3 2 1) (1 2 3) (2 1 3) (2 3 1))  
(list-lex-less? '(1 2) '(1 2)) => #t  
(list-lex-less? '(1 2) '(1 2 3)) => #t  
(list-lex-less? '(2) '(1 3)) => #f  
(list-lex-less? '(1 2 3) '(1 2)) => #f  
(my-sort '(2 3 1) <) => '(1 2 3)  
(my-sort '((3) (1 2 3) (1 3)) list-lex-less?) => '((1 2 3) (1 3) (3))  
(sorted-permutations '(1 3 2)) => '((1 2 3) (1 3 2) (2 1 3) (2 3 1) (3 1 2) (3 2 1))
```

You are required to provide pure functional implementations from scratch, that do not employ advanced functions or imperative features. Therefore, you are allowed to use *only* the following basic Scheme functional constructs:

- function creation: `lambda`
- binding: `define`, `let`, `let*`, `letrec`
- booleans: `not`, `and`, `or`
- conditionals: `if`, `cond`
- basic list operations: `car`, `cdr`, `cons`, `list`, `append`, `null?`
- mapping: `map`, `apply`

Submit your code as a file `sorted-permutations.rkt`.

**READ ME!** Submit your answers as a *single pdf file* in OWL. Solutions should be typed but readable (by others!) hand-written solutions are acceptable. Source code, if required, is submitted as separate files.

**L<sup>A</sup>T<sub>E</sub>X:** For those interested, the best program for scientific writing is L<sup>A</sup>T<sub>E</sub>X. It is far superior to all the other programs, it is free, and you can start using it in minutes; here is an introduction: <https://tobi.oetiker.ch/lshort/lshort.pdf>