**CS3342 – Assignment 3**
**due Mar. 26, 2023**
**2-day no-penalty extension until: Mar. 28, 11:55pm**

1. (10pt) Explain how you can use short-circuiting for the evaluation of the XOR logical operator.

2. (20pt) Consider the general form of a `while` loop:

   ```
   before_loop
   while ( condition ) do
       body
   after_loop
   ```

   Translate this code into recursive code. That is, you need to write a piece of (peudo)code that performs the same task as the `while` loop, but without any iterative loops (such as `for, while, do..until, do..while`), instead using only recursion. This is a rather informal exercise. No rigorous notation or proof is required. Only the correct idea.

3. (20pt) Consider the following definitions:

   - $\texttt{T} \equiv \lambda pq.p$
   - $\texttt{F} \equiv \lambda pq.q$
   - $\texttt{NOT} \equiv \lambda pqr.p\,r\,q$
   - $\texttt{XOR} \equiv \lambda pq.p\,(\texttt{NOT}\,q)\,q$

   (a) (14pt) Reduce, using both call-by-value and call-by-name reduction (two separate reductions):

   $$\texttt{XOR}\ p\,(\texttt{NOT}\,p)$$

   For all computations you perform, indicate clearly the reduction being done by underlying the abstraction used and the argument it is applied to: $(\underline{\lambda x}.M)\underline{N}$.

   *Note:* You are allowed to use online lambda expression calculators, but not to submit screen shots. And make sure to understand what they do.

   (b) (6pt) Does your computation indicate that `XOR` is behaving as the exclusive-or operator you know?

4. (50pt) Write a Scheme program, `inversion_count.rkt`, that takes a permutation as input and outputs the number of inversions. Here are some examples:

   ```
   (count-inversions '()) => 0
   (count-inversions '(1 2 3)) => 0
   (count-inversions '(1 3 2)) => 1
   (count-inversions '(3 2 1)) => 3
   ```

   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`

**READ ME!** Submit source code, if required, as separate file(s).

Submit your answers as a single pdf file in OWL. Solutions should be typed; readable (by others!) hand-written solutions are also acceptable.

**External tools:** You are allowed to use any external tools, such as JFLAP, ChatGPT, lambda expression calculators, etc., to help you solve the assignments. Make sure you understand the solutions as no tools will be available during the exams!

**LaTeX:** For those interested, the best (the only!) program for scientific writing is LaTeX. It is free and you can start using it in minutes: `https://tobi.oetiker.ch/lshort/lshort.pdf`