Specification for Assignment 4 of 4

*Your submission for this assignment **must include your full name** (as it appears on cuLearn) and you nine-digit **student number as a comment** at the top of **every source file you submit**.*

*Your submission for question 2 is not code and must be a **single pdf file** with a **file name of 'comp3007_w19_#########_a4_2.pdf'** (with the number signs replaced by your nine-digit student number). It **must be written** using **Microsoft Word**, **Google Docs**, or **LaTeX**. **Photographs or scans** of handwritten submissions **will not be accepted**.*

*Your submissions for questions 1 and 3 will include both Haskell and Prolog and must be named **'comp3007_w19_#########_a4_1a.hs'**, **'comp3007_w19_#########_a4_1b.pl'**, and **'comp3007_w19_#########_a4_3.pl'**.*

Compress your submission** into a zip archive named **'comp3007_w19_#########_a4.zip'

***Late assignments will not be accepted** and will **receive a mark of 0**.*

***Submissions that crash** (i.e., terminate with an error) on execution will **receive a mark of 0**.*

The due date for this assignment is Saturday, April 6, 2019, by 11:00pm.

Question 1: "Counting Zerofrees"

A function to count the number of elements in a list is relatively straightforward in either Haskell or Prolog, and for this question you will count the number of elements in a list of integers that are considered "Zerofree". (n.b., A "Zerofree" number does not contain the digit 0 when written in decimal; e.g., 8, 9, 11, 12, etc. are "Zerofree", but 10 is not.)

In Haskell, your function to count the number of "Zerofree" elements would have only one parameters (i.e., the list of integers) and would return a single value (the number of elements within that range). In Prolog, it would be a predicate of arity two that succeeds if and only if the second element is the number of elements in the first list argument that are "Zerofree" (and you would need to be able to leave the second element uninstantiated and have Prolog count it for you).

```
HASKELL > countZerofree [997, 998, 999, 1000, 1001]
3

PROLOG ?- countZerofree ([997, 998, 999, 1000, 1001], X).
X = 3 ;
false.
```

n.b., The specification for this question continues on the following page.

For this question, you will write this function twice - once using Haskell and once using Prolog. Both solutions must be recursive, and the countZerofree function itself in the Haskell solution must be tail-call optimized. Neither solution may use higher order functions.

Obviously, you will need to start by writing a function to test whether an integer is "Zerofree" or not, and this must be a recursive function. Furthermore, as has been the case for most of our assignments, you may not use any built-in functions to complete your submission. This means that you will need to write your own implementations of basic functions like `length`, `mod`, `rem`, `take`, `drop`, and `reverse`, should you wish to use them. You are, however, free to use operators, like the list constructor operator (`:`), the list concatenation operator (`++`), and the arithmetic operators, etc. to complete this assignment.

Question 2: "Best Behaviour"

Use structural induction to prove `(length x) - (length (filterPQ x)) = (countIf x)`. Use the following implementations for `myLength`, `filterPQ`, `beforeP`, `afterQ`, and `countIf` and refer to the individual lines in these implementations using the labels `L1, L2, F1, F2A, F2B, F2C, B1, A1, C1, C2A, C2B,` and `C2C`. You must follow the process of structural induction as it was demonstrated in class and you must show all your work (including the line applied during each step of your equational reasoning).

```
        myLength :: [Char] -> Int
L1      myLength [] = 0
L2      myLength (h:t) = 1 + (myLength t)

        filterPQ :: [Char] -> [Char]
F1      filterPQ [] = []
        filterPQ (h:t)
F2A       | beforeP h = h : (filterPQ t)
F2B       | afterQ h = h : filterPQ t
F2C       | otherwise = filterPQ t

        beforeP :: [Char] -> Bool
B1      beforeP x = ord x < 80

        afterQ :: [Char] -> Bool
A1      afterQ x = ord x > 81

        countIf :: [Char] -> Char
C1      countIf [] = 0
        countIf (h:t)
C2A       | h == "P" = 1 + (countIf t)
C2B       | h == "Q" = 1 + (countIf t)
C2C       | otherwise = countIf t
```

Question 3: "Ancient History"

For this question, you must write a Prolog program that will solve the following logic puzzle by modeling it as a finite state machine. Your solution to this question MUST follow the approach we demonstrated in class for solving this type of logic puzzle with a finite state machine. Additionally, your solution must answer True or False for the first question (i.e., Is it possible...?) and it must provide an unambiguous description of the action plan for the second question (i.e., How?). If more than one solution is possible, your program must provide all solutions.

A group of three humans, one neanderthal, and his two pet saber-toothed tigers have found themselves on the west bank of a river, and they need to cross to the east side using a small rowboat. The rowboat only has enough room for two entities at any given time, and only the neanderthal or one of the humans is capable of actually rowing the boat from one bank of the river to the other. It is also necessary, at all times, for the number of humans on either side of the river to be greater than or equal to the number of saber-toothed tigers (to ensure the tigers do not eat the humans).

Is it possible for all of them to cross the river in 15 moves or less? How?