CS115 – Lab 5 – Deconstructing and Constructing Lists

Spring 2018

1 Question 1

Write a function (factor-both? L a b) that consumes a (listof Nat), a Nat, and a Nat. It returns the list containing all the values in L that are divisible by both a and b.

(factor-both? (list 17 25 30 33 45) 3 5) => (list 30 45)

Write a function (factor-either? L a b) that consumes a (listof Nat), a Nat, and a Nat. It returns the list containing all the values in L that are divisible by either a or b.

(factor-either? (list 17 25 30 33 45) 3 5) => (list 25 30 33 45)

2 Question 2

A leap year is a year that is exactly divisible by four, except for years that are exactly divisible by 100, unless it is also divisible by 400.

Write a function (leap-year? year) that determines whether year is a leap year.

(leap-year? 2004) => #true
(leap-year? 2001) => #false

Don’t use cond for this question.

Write a function (keep-leapyears L) that consumes a (listof Nat), representing a set of years. The function returns the list of years that are leap years.

(keep-leapyears (list 1900 2000 2004)) => (list 2000 2004)

3 Question 3

In Chemistry, we can classify a solution by its pH (generally between 0 and 14):

Acid  pH < 6.6
Neutral  6.6 ≤ pH ≤ 7.3
Base  pH > 7.3
Write a function `(classify-by-pH lop type)` that consumes a `(listof Num)` and a `Str`. `lop` represents the pH of some solutions. `type` is one of "acid", "neutral" or "base".

The function returns a list containing all the pH values in the appropriate range.

For example:

- `(classify-by-pH (list 4.2 3.8 6.6 7.0 7.3) "acid") => (list 4.2 3.8)
- `(classify-by-pH (list 4.2 3.8 6.6 7.0 7.3) "neutral") => (list 6.6 7.0 7.3)
- `(classify-by-pH (list 4.2 3.8 6.6 7.0 7.3) "base") => '()"

4 Question 4

You are given the following code:

```scheme
;; (cube-threes L) cube each value in L that is divisible by 3
;; cube-threes: (listof Num) -> (listof Num)
;; Example:
(check-expect (cube-threes (list 0 2 3 4 5 6 7)) (list 0 27 125))

(define (cube-threes L)
  (map (lambda (x) (* x x x))
    (filter (lambda (n) (= 0 (remainder n 3))) L)))
```

Ex. Rewrite `cube-threes` so it does not use `map` or `filter`. Use `foldr` only once.