Question 1: Phun with pH

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

- **Acid** $\text{pH} < 6.6$
- **Neutral** $6.6 \leq \text{pH} \leq 7.3$
- **Base** $\text{pH} > 7.3$

Write a function `(classify-by-pH L type)` that consumes a `(listof Num)` and a `Str`. `L` 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 interval.

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") => '()`

Remember to define appropriate constants.

Question 2: Positive Differences

If we want the answer to be a natural number $0, 1, 2, 3, \ldots$, then we can’t subtract a bigger number from a smaller number.

Write a function `(good-differences L)` that consumes a `(listof (list Nat Nat))`, and returns all the items where the second value is no larger than that first.

- `(good-differences (list (list 7 5) (list 9 9) (list 3 4))) => (list (list 7 5) (list 9 9))`

Write a function `(sum-good-differences L)` that returns the sum of all the differences which are natural numbers.

- `(sum-good-differences (list (list 7 5) (list 9 9) (list 3 4))) => 2`
- `(sum-good-differences (list (list 7 5) (list 3 4) (list 9 3) (list 17 16))) => 9`

Question 3: Cubes

You are given the following code:

```scheme
;;; (div3? n) return #true if n is divisible by 3, otherwise #false.
(define (div3? n) (= 0 (remainder n 3)))

;;; (cube x) return the cube of x.
(define (cube x) (* x x x))

;;; (cube-threes L) cube each value in L that is divisible by 3
;;; cube-threes: (listof Int) -> (listof Int)
```

```scheme
```
Example:
(check-expect (cube-threes (list 0 2 3 4 5 6 7)) (list 0 27 216))

(define (cube-threes L)
  (map cube (filter div3? L)))

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

Question 4: Leap Years

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 (leap-year? 1900) => #false (leap-year? 1904) => #true

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)