# CS380: Modern Functional Programming
**Prof. Richard Eisenberg**
**Spring 2017**

## Lists

Use list comprehensions to solve these problems. They are taken from `http://projecteuler.net`; they are problems 1 and 2.

The following definitions from the Haskell standard library will be helpful:

```
  -- sum xs calculates the sum of all the elements in xs. Thus, sum [1, 2, 3] equals 6.
sum :: Num a ⇒ [a] → a
  -- zipWith f xs ys applies the function f to corresponding elements in xs and ys.
  -- Thus, zipWith (−) [4, 7, 10] [1, 2, 3] equals [3, 5, 7]. If one list is longer than the
  -- other, the extra elements in the longer list are ignored.
zipWith :: (a → b → c) → [a] → [b] → [c]
  -- tail xs removes the first element from the list xs and returns the remainder.
  -- Thus, tail [1, 2, 3] equals [2, 3]. tail errors if the list is empty.
tail :: [a] → [a]
  -- even x is a predicate that determines whether or not x is even. Thus,
  -- even 4 is True while even 9 is False.
even :: Integral a ⇒ a → Bool
  -- takeWhile f xs returns the longest prefix of the list xs such that every
  -- element in the prefix satisfies predicate f. Thus, takeWhile (<3) [1, 2, 3, 4, 1, 2, 3]
  -- equals [1, 2].
takeWhile :: (a → Bool) → [a] → [a]
```

1. If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.

   Write an expression that evaluates to the sum of all the multiples of 3 or 5 below 1000.

2. Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be: $1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$

   We can define the infinite list of Fibonacci numbers in Haskell with

   ```
   fibs :: [Integer]
   fibs = 0 : 1 : zipWith (+) fibs (tail fibs)
   ```

   By considering the terms in the Fibonacci sequence whose values do not exceed four million, write an expression that evaluates to the sum of the even-valued terms.