1. (Exercise 1.33 in text) Design a procedure filter-accumulate that further generalizes the accumulate procedure that was presented in class.

   (define filter-accumulate (lambda (filter combiner null-value f next a-start a-last)

   The parameter filter will be passed a predicate function of one argument that returns true or false, depending on whether its argument satisfies the predicate. The other parameters of filter-accumulate are the same as the parameters of accumulate that we discussed in class. filter-accumulate will combine only those terms derived from values in the range that satisfy the predicate given by parameter filter.

2. Write a single unconditional expression that calls filter-accumulate to compute the sum of the squares of every integer between 28 and 63 inclusive where at least one of the integer’s digits is even. So the result will be

   \[ 28^2 + 29^2 + 30^2 + 32^2 + 34^2 + 36^2 + 38^2 + 40^2 + 41^2 + \ldots \]

3. In order to test your expression in the lab submission system, replace <your-expression-1> in procedure test1 with your expression above that calls filter-accumulate

   (define test1 (lambda ()
      <your-expression-1>)
   ))

4. The following procedure returns true if its argument is a prime.

   (define prime? (lambda (n)
      ; return true if n is a prime number

      (define smallest-divisor (lambda(n)
         ; find the smallest divisor of n
         (define find-divisor (lambda (n test-divisor)
            ; find the smallest divisor of n
            ; starting with test-divisor
            (cond ((> (* test-divisor test-divisor) n) n)
                  ((= (remainder n test-divisor) 0)
                   test-divisor)
                  (else (find-divisor n (+ test-divisor 1)))))
         (find-divisor n 2))
      (= n (smallest-divisor n))))
You will need to use this procedure, and you can load it from ~carberry/test-prime.scm

Write a single unconditional expression that calls filter-accumulate to compute the sum of the cubes of the prime numbers in the interval 4 to 45 inclusive.

5. In order to test your expression in the lab submission system, replace <your-expression-2> in procedure test2 with your expression above that calls filter-accumulate

(define test2 (lambda ()
  <your-expression-2>
))

6. (Exercise 1.42) Define a procedure compose that has two parameters g and f (in that order, where g computes a function of one argument and f computes a function of two arguments), and returns a procedure that computes g(f(x,y)) — ie., the composition of g and f. Thus

   ((compose square max) 4 7)    returns 49
   ((compose square max) 5 3)    returns 25
   ((compose cube min) 3 2)      returns 8
   ((compose square (lambda(x y) (remainder x y))) 56 6)  returns 4

7. Copy the procedures for filter-accumulate, test1, test2, and compose into the lab submission system, and submit your work for Homework-3.