Tutorial Problem Set 1

CS 152 – Abstractions and Paradigms in Programming

Date - 19th Jan 2017 Submission Date: 26th Jan 2017

1. A quick question to wind up RSA encoding. Suppose that the RSA algorithm uses the prime numbers p and q to generate the encryption, the encoding exponent is e and the code of a message (a number) is c. Assume that e is a prime that does not divide $(p-1)(q-1)$. Write a function `(decode p q e c)` to decode the message.

2. Suppose we wanted to write the following functions, explained through examples of how they may be called:

   (a) `(f0 1 100)`: Starting with 1, the sum of every 13th number in the interval 1 to 100.

   (b) `(f1 2 100)`: The sum of squares of all odd numbers between 2 and 100.

   (c) `(f2 1 100)`: The product of the factorials of multiples of 3 between 1 and 100.

   (d) `(f3 10 200)`: The sum of the squares of the prime numbers in the interval 10 to 200.

   (e) `(f4 50)`: The product of all positive integers less than `50` that are relatively prime to 50.

   Write a higher-order function called `filtered-accumulate` so that the functions described above are instances of `filtered-accumulate`. Write the functions `f0, f1, f2, f3` and `f4` in terms of `filtered-accumulate`. You can assume that the arguments to `f0, f1, f2, f3` and `f4` are positive.

3. Simpson's rule is an accurate method of numerical integration. Using Simpson's rule, the integral of a function f between a and b is approximated as

   $$h/3 * [y_0 + 4 * y_1 + 2 * y_2 + 4 * y_3 + 2 * y_4 + ... + 4 * y_{n-1} + y_n]$$

   where $h = (b-a)/n$, for some even integer n, and $y_k = f(a+kh)$. (Increasing n increases the accuracy of the approximation.)

   Define a procedure `(simpson f a b n)` that returns the value of the integral, computed using Simpson's Rule.

4. Assume that we represent a rational number p/q as $(cons\ p\ q)$. With this representation, define the following functions on rational numbers.

   (a) `(simplify r)` - converts a rational number r into its simplest form. $12/18$ simplifies to $2/3$.

   (b) `(add r1 r2)` - adds two rational numbers r_1 and r_2.

(c) `(muliply r1 r2)` - multiplies two rational numbers.

(d) `(divide r1 r2)` - divides r1 by r2.

5. Write a function `(minchange n)` which will return the minimum number of coins required to give change of n paise. Assume that you have sufficient numbers of 1p, 2p, 3p, 5p, 10p, 20p, 25p and 50p coins. What is the highest value of n that you can go upto in 5 minutes of CPU time. Read the simple solution from SICP and try to improve upon the solution.

6. Monte Carlo integration: Consider computing the area of a region of space described by a predicate $P(x, y)$ that is true for points (x, y) in the region and false for points not in the region. For example, the region contained within a circle of radius 3 centered at $(5, 7)$ is described by the predicate that tests whether $(x - 5)^2 + (y - 7)^2 \leq 3^2$. To estimate the area of the region described by such a predicate, begin by choosing a rectangle that contains the region. For example, a rectangle with diagonally opposite corners at $(2, 4)$ and $(8, 10)$ contains the circle above. The desired integral is the area of that portion of the rectangle that lies in the region. We can estimate the integral by picking, at random, points (x, y) that lie in the rectangle, and testing $P(x, y)$ for each point to determine whether the point lies in the region. If we try this with many points, then the fraction of points that fall in the region should give an estimate of the proportion of the rectangle that lies in the region. Hence, multiplying this fraction by the area of the entire rectangle should produce an estimate of the integral. Implement Monte Carlo integration as a procedure estimateintegral that takes as arguments a predicate P, upper and lower bounds x_1, x_2, y_1, and y_2 for the rectangle, and the number of trials to perform in order to produce the estimate. Use your estimate-integral to produce an estimate of π.

   Use the function defined below to generate random numbers between x_1 and x_2.

```
(define (scaled-random x1 x2)
  (+ x1 (* (random)
           (- x2 x1))))
```

7. You all know what Fibonacci numbers are: They are given by the sequence: $fib(0) = 1, fib(1) = 1, and\ fib(n) = fib(n-1) + fib(n-2)$.

   (a) Write a tail recursive version of fib called `fib-tr`.

   (b) There is a nice way to compute $fib(n)$ even faster than the tail recursive version. We can write fib as a matrix as:

$$\begin{pmatrix} fib(n) \\ fib(n-1) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} fib(n-1) \\ fib(n-2) \end{pmatrix}$$

   Therefore, a closed form version of the equation will be

$$\begin{pmatrix} fib(n) \\ fib(n-1) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

   And you know how to do exponentiation very fast, don't you? Using this idea, write a version of fib called `fib-lightning`.