MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.001—Structure and Interpretation of Computer Programs
Fall Semester, 1997

Problem Set 2 Solutions

Pre-Lab Exercise 2.1 The following procedures evaluate continued fraction expressions generating recursive and iterative processes respectively.

```
(define (cont-frac-r n d k)
  (define (helper i)
    (if (> i k)
        0
        (/ (n i)
           (+ (d i)
              (helper (1+ i))))))
  (helper 1))
(define (cont-frac-i n d k)
  (define (iter result i)
    (if (= i 0)
        result
        (iter (/ (n i)
                 (+ (d i) result))
              (-1+ i))))
  (iter 0 k))
```

To approximate $1/\phi \approx 0.618034$ with 4 decimal place accuracy requires 12 terms, yielding the result 0.618025.

Lab Exercise 2.2 Brouncker's method for computing π alternately yields approximations that alternately are larger and smaller than the correct value. Both errors are smaller than 0.005 if at least 199 terms are used.

Pre-Lab Exercise 2.3 The following procedure evaluates the arctangent function using its continued fraction expansion.

```
(define (atan-cf k x)
  (cont-frac-i (lambda (n)
                 (if (= n 1)
                     x
                     (square (* (-1+ n) x))))
               (lambda (n)
                 (-1+ (* 2 n)))
               k))
```

Lab Exercise 2.4 Three decimal places of accuracy (error less than 0.0005) can be achieved with 1, 5, 13, 44, 131, 437 terms for arguments of 0, 1, 3, 10, 30, and 100 respectively. Roughly speaking, the number of terms is proportional to the size of the argument.

Pre-Lab Exercise 2.5 One definition of a procedure that computes the value of a nested accumulation:

```
(define (nested-acc op r term k)
  (define (helper i)
    (if (= i k)
        ((op i) (term i) r)
        ((op i) (term i) (helper (1+ i)))))
  (helper 1))
```

Lab Exercise 2.6 The following procedure computes an approximation to $f(x)$

```
(define (f x k)
  (nested-acc (lambda (n)
                (lambda (a b)
                  (sqrt (+ a b))))
              0
              (lambda (n) x)
              k))
```

The error is less than 0.0005 if 8 terms are used.

Lab Exercise 2.7 The following procedure computes the sum of the first `k` terms of the Taylor series for the sine function using Horner's representation:

$$\sin x = x \times \left(1 - \frac{x^2}{3 \times 2} \times \left(1 - \frac{x^2}{5 \times 4} \times \left(1 - \frac{x^2}{7 \times 6} \cdots \right)\right)\right)$$

The `sine-taylor` procedure uses the `nested-acc` procedure to compute $2k - 1$ terms of a nested expression in which the operators are alternately `-` and `*`, the residual is always unity, and the nth term is alternately 1 and $x^2/n(n+1)$.

```
(define (sine-taylor k x)
  (define the-ops
    (lambda (n)
      (if (odd? n) - *)))
  (let ((x-squared (* x x)))
    (define the-terms
      (lambda (n)
        (if (odd? n)
            1
            (/ x-squared  (* (+ n 1) n)))))
    (* x
       (nested-acc
        the-ops
        0
        the-terms
        (- (* 2 k) 1)))))
```

Evaluating `(sine-taylor 4 1)`, for example, returns 0.0998334, which is within 0.000003 of the value returned by Scheme's own `sin` procedure.

Pre-Lab Exercise 2.8 The following is one definition of the `repeated-build` procedure.

```
(define (repeated-build k n d r)
  ((repeated
    (lambda (x)
      (build n d x))
    k)
   r))
```

Lab Exercise 2.9 A k-term approximation to the reciprocal of the golden ratio can be computed by evaluating `(repeated-build k 1 1 1)`.

Lab Exercise 2.10 Applying procedure `r` creates a procedure that computes $r_k(x)$.

```
(define (r k)
  (lambda (x)
    (repeated-build k 1 1 x)))
```