**Problem 1**

(define (H n)

(if (= n 0)

0

(+ (/ 1 n) (H (- n 1) ) )

)

)

(define (E-constant x) (abs (- (H x) (log x) ) ) )

(E-constant 10)

0.6263831609742079

(E-constant 100)

0.5822073316515288

(E-constant 1000)

0.5777155815682082

**Problem 2**

(define (prime? n)

(define (divisor? k) (= 0 (modulo n k) ) )

(define (divisors-upto k)

(and (> k 1)

(or (divisor? k) (divisors-upto (- k 1) ) ) ) )

(not (divisors-upto (- n 1) ) ) )

(define (count-primes t)

(define (count-primes-helper t)

(cond ( (prime? t ) 1)

(else 0) )

)

(if (= t 1)

1

(+ (count-primes-helper t )(count-primes (- t 1) ) ) ) )

(count-primes 11)

6

(count-primes 5)

4

**Problem 3**

**A**

(define (Lucas n)

(cond ((= n 0) 2)

( (= n 1) 1)

( (> n 1) (+ ( Lucas (- n 1) )

( Lucas (- n 2) ) ) )

)

)

(Lucas 5)

11

**B**

(define (Lucas-ratio n) (- (Lucas n) (Lucas (- n 1) ) ) )

(Lucas-ratio 20)

#e1.6180340143330837522729703...

(Lucas-ratio 21)

#e1.6180339789779863819660210...

(Lucas-ratio 22)

#e1.6180339924824317698970420...

(Lucas-ratio 23)

#e1.6180339873241926116708330...

;The ratios of Lucas number looks strikingly familiar. It seems as if they don’t change at all.

;Fibonacci number

(define (Fibonacci n)

(cond ( (= n 0) 0)

( (= n 1) 1)

( (> n 1) (+ ( Fibonacci (- n 1) )

( Fibonacci (- n 2) ) ) )

)

)

(define (Fibonacci-ratio n) (- (Fibonacci n) (Fibonacci (- n 1) ) ) )

(Fibonacci-ratio 20)

#e1.6180339631667065295383879...

(Fibonacci-ratio 21)

#e1.61803399852

(Fibonacci-ratio 22)

#e1.6180339850173579389731408...

(Fibonacci-ratio 23)

#e1.6180339901755970865563773...

;The Fibonacci ratio is very similar and close to the Lucas ratio, if not the same.

**C**

;Letting SCHEME do the 30th, 35th, and 40th value of Lucas number, I suspect it will take very long to return result for 50th value, as it takes quite awhile for 45th to be computed

(define (fast-Lucas-help n k luc-a luc-b)

(if (= n k)

luc-a

(fast-Lucas-help n (+ k 1) (+ luc-a luc-b) luc-a) ) )

(define (fast-Lucas n) (fast-Lucas-help n 1 1 2) )

|  |  |  |
| --- | --- | --- |
|  | Rercursive calls made by (Lucas k) | Recursive calls made by (fast-Lucas-help k 1 1 2) |
| K = 1 | 0 | 0 |
| K = 2 | 2 | 1 |
| K = 3 | 3 | 2 |
| K = 4 | 4 | 3 |
| K = 5 | 5 | 4 |
| K = 6 | 6 | 5 |

**Problem 4**

**A**

(define (golden n)

(cond ( (= n 1) 2)

( ( > n 1) (+ 1 (/ 1 (golden (- n 1) ) ) ) )

)

)

(golden 1)

2

(golden 2)

3/2

(golden 3)

5/3

**B**

(define (golden-new n)

(cond ( (= n 1) 1)

( (> n 1) (sqrt (+ 1 (golden-new (- n 1) ) ) ) )

)

)

(golden-new 3)

1.5537739740300374

(golden-new 4)

1.5980531824786175

(golden-new 5)

1.6118477541252516

**Problem 5**

(define (one-sample)

(define (square x) (\* x x) )

(let ( (x (- ( \* 2 (random) ) 1) )

(y (- ( \* 2 (random) ) 1) ) )

(if (<= (+ (square x)

(square y) )

1)

1

0) ) )

(define (pi-samples k)

(if (= k 1)

(one-sample)

(+ (one-sample) (pi-samples (- k 1) )

) )

)

(define (pi-estimate n)

(define (p n) (/ (pi-samples n) n) )

(\* 4 (p n) ) )

(pi-estimate 5)

3.2

**Problem 6**

(define (interval-sum m n)

(if (= m n)

m

(+ m (interval-sum (- m 1) n) ) ) )

(interval-sum 20 15)

105

\*In this version, m is higher than n so it must reduce itself by 1 to create solution.

(define (interval-sum m n)

(if (= m n)

m

(+ n (interval-sum m (- n 1) ) ) ) )

(interval-sum 15 20)

105

\*In this version, n is higher than m so it must reduce itself by 1 to create solution.

(define (interval-sum m n)

(if (= m n)

m

(+ m

(interval-sum (+ m 1) (- n 1) )

n) ) )

(interval-sum 7 10)

🡪This pair of number doesn’t work because the difference between them is an odd number. Since one number is added to 1 while the other is taken out 1 unit, the difference between each number must be an even number for this to work. Other the difference is 0. In addition m must be less than n for this function to work, or else m and n will never be equal to each other, and SCHEME will run in an infinite loop and run out of memory.