Code of factorial

1029-24-9540 Yamazaki Keitaro

October 19, 2012

1 Iterative Factorial

```
(define fact-iter (lambda (n)
        (define iter (lambda (prod count)
                (if (> count n)
                        prod
                        (iter (* prod count) (+ count 1))
                )
        ))
        (iter 1 1)
))
```

Run:
(fact-iter 100)

Result:

```
933262154439441526816992388562667004907159682643816214685929638
952175999932299156089414639761565182862536979208272237582511852
109168640000000000000000000000000
```

Description:
This function calculates factorial not recursively but iteratively.
So the space complexity is smaller than recursive factorial.

2 Recursive Fibonacci

```
(define fib (lambda (n)
        (cond
                ((= n 0) 0)
                ((= n 1) 1)
                (else (+
                        (fib (- n 1))
                        (fib (- n 2))
                ))
        )
))

(define fib-i (lambda (n)
        (define iter (lambda (a b count)
                (if (= count n)
                        a
                        (iter b (+ a b) (+ count 1))
                )
        ))
        (iter 0 1 0)
))
```

Run:

(fib 10)
(fib 20)
(fib 30)
(fib-i 10)
(fib-i 20)
(fib-i 30)

Result:

(fib 10) => 55
(fib 20) => 6765
(fib 30) => 832040
(fib-i 10) => 55
(fib-i 20) => 6765
(fib-i 30) => 832040

Description:

Function ”fib” recursively calculates fibonacci, and function ”fib-i” iteratively calculates fibonacci.
Computational complexity of function ”fib” is exponential but that of function ”fib-i” is O(n).
So function ”fib-i” needs less caluculation.