Metody Programowania lista 2

Szymon Kopa

20 lutego 2018

1 Zadanie 2

```
#lang racket

(define (compose f g)
  (lambda (a) (f (g a))))
(
(define (square x) (* x x))

(define (inc x) (+ x 1))

(define (identity x) x)
```

2 Zadanie 3

```
#lang racket

(define (repeated f n)
  (cond
    [(= n 0) identity]
    [(= n 1) f]
    [else (compose (identity f) (repeated f (- n 1 )))]))
```

3 Zadanie 5

```
#lang racket

(define (accumulate combiner null-value term a next b)
  (define ( accumulate-iter a acc)
    (if [> a b ]
        null-value
        (accumulate-iter (next a) (combiner acc (term a))))))

;REKURENCJA

(define (accumulate-recursive combiner null-value term a next b)
  (if [> a b]
      null-value
      (combiner (term a) (accumulate-recursive term (next a) next b))))
```

4 Zadanie 6

```
#lang racket

(define (cont-frac num den k)
```

```
  (define (counter acc)
    (cond
      [(= acc k) 0]
      [else (/ (num acc) (+(den acc) counter )(+ acc 1)])))
  (counter 0))

;iter
(define (cont-frac num den k)
  (define (counter acc wynik)
    (cond
      [(= acc 0) wynik]
      [else (counter (\ (num acc) (+(den acc) wynik))(- acc 1))])))
```