**1. A function (binomial N k) that returns the binomial coefficients C(N, k), defined recursively as: C(N,0) = 1, C(N, N) = 1, and, for 0<k < N, C(N, k) = C(N-1, k) + C(N - 1, k -1).**

* **Test binomial for C(4,0), C(8, 8), C(3,2) and C(7, 4).**

(define (binomial N k)

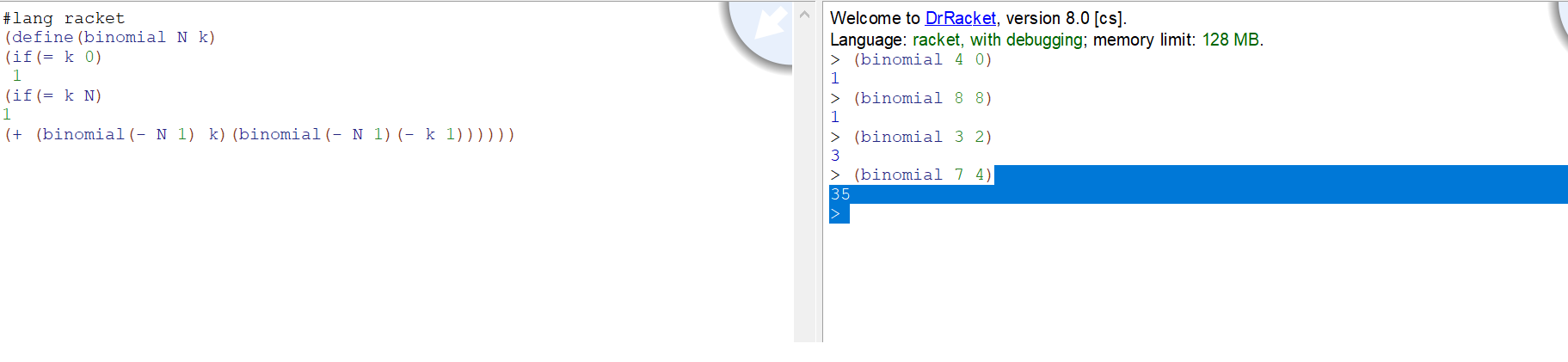
(if (= k 0)

1

(if (= k N)

1

(+ (binomial (- N 1) k) (binomial (- N 1) (- k 1))))))

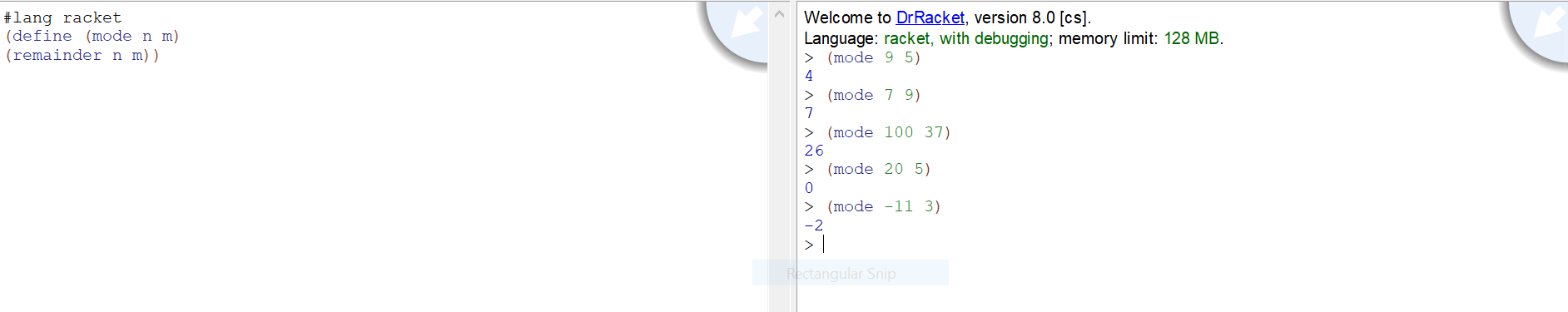
****

**2. A function (mod N M) that returns the modulus remainder when dividing N by M.**

* **Test mod for arguments 9 and 5, 7 and 9, 100 and 37, 20 and 5, -11 and 3.**

(define (mode n m)

(remainder n m))



**3. A function (binaryToDecimal b) that takes a binary number and returns its decimal value.(binaryToDecimal 1101) returns 13.**

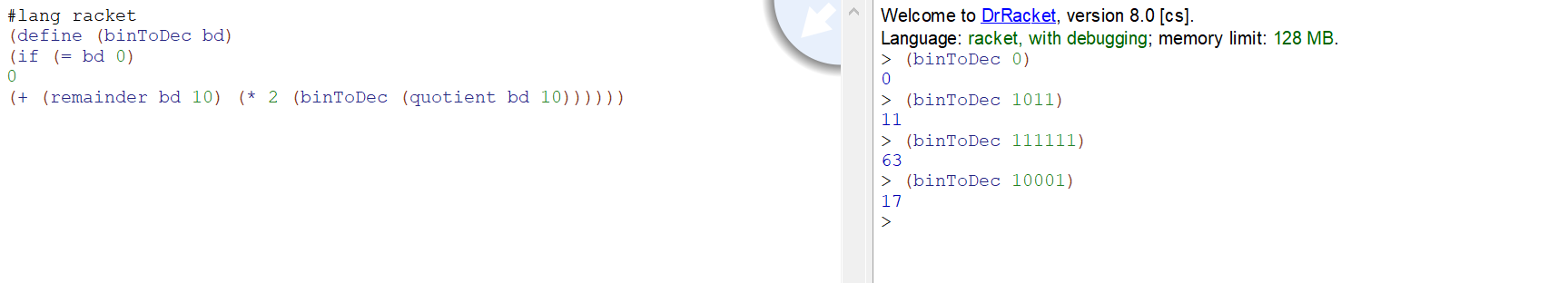
* **Test binaryToDecimal with arguments 0, 1011, 111111, 10001.**

(define (binToDec bd)

(if (= bd 0)

0

(+ (remainder bd 10) (\* 2 (binToDec (quotient bd 10))))))



**4. A function (addBinary binaryList) that takes a list of binary numbers and returns their decimal sum. (addBinary '(1101 111 10 101)) returns 27**

* **Test addBinary with (1101 111 10 101), (0), (11011).**

(define (binaryToDecimal b)

(if (= b 0)

0

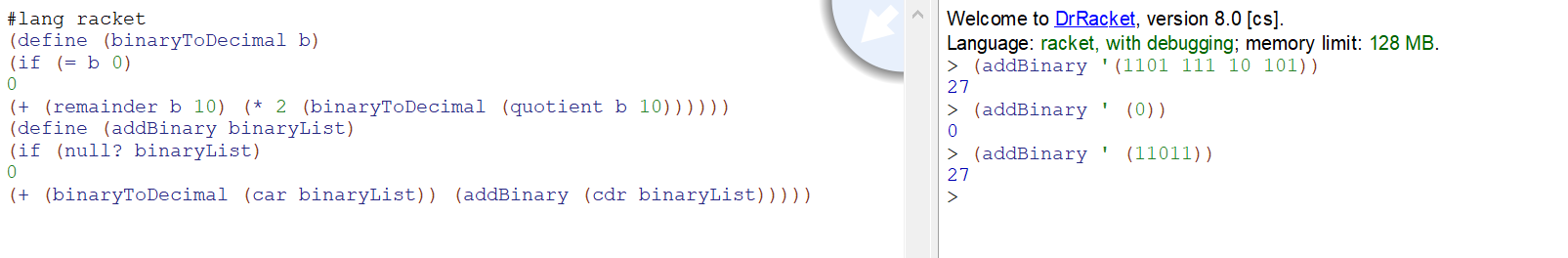
(+ (remainder b 10) (\* 2 (binaryToDecimal (quotient b 10))))))

(define (addBinary binaryList)

(if (null? binaryList)

0

(+ (binaryToDecimal (car binaryList)) (addBinary (cdr binaryList)))))



**5. A function (min list) that returns the smallest value in a simple list of integers**

* **Test min with (4 5 1 2 5), (3), (), (5 5 5)**

(define (min list)

(cond

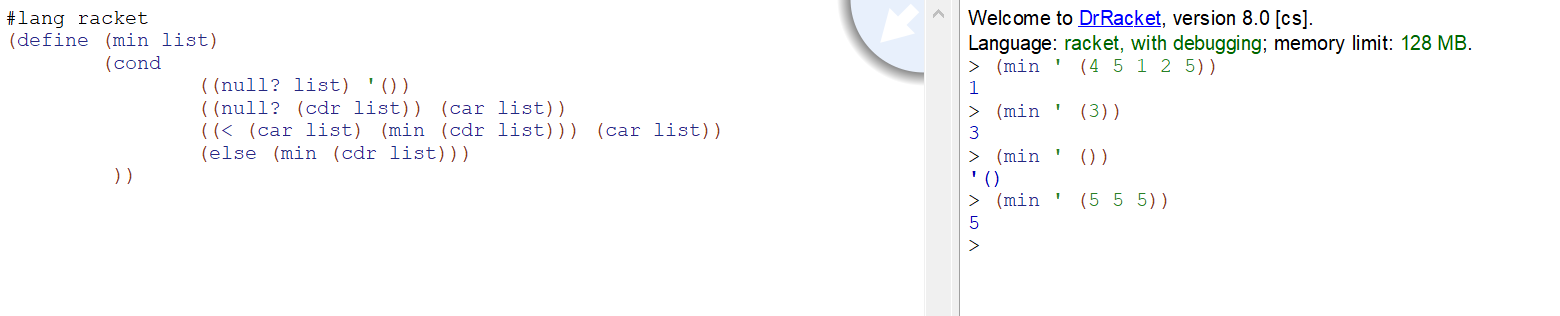
((null? list) '())

((null? (cdr list)) (car list))

((< (car list) (min (cdr list))) (car list))

(else (min (cdr list)))

))



**6. A function (myRemove atm list) that removes all occurrences of the atom atm from a simple list, returning list with atm removed. myRemove should return the original list if atm is not found.**

* **Test myRemove with atom a and list arguments (), (a), (a b c d a b a a), (x y z), (a (x y z) (r s t a)), (((a (l a) b) a) m a).**

(define (myRemove atm lst clean)

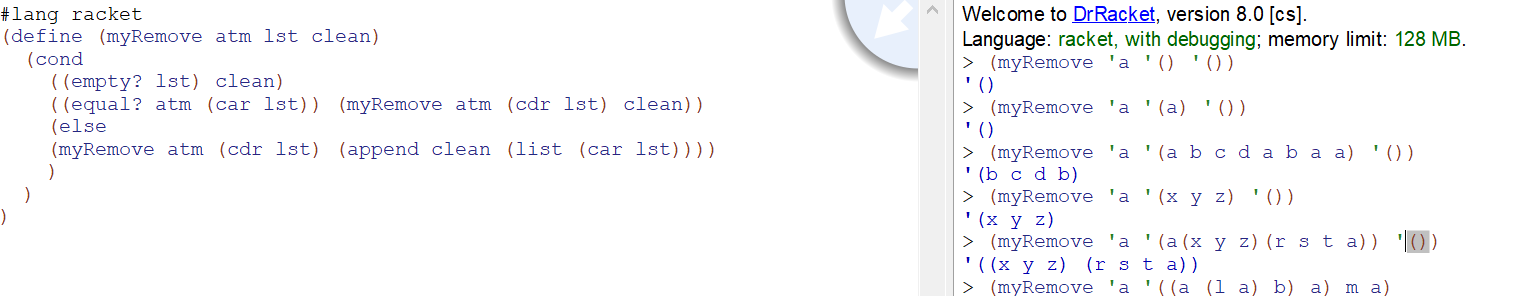
(cond

((empty? lst) clean)

((equal? atm (car lst)) (myRemove atm (cdr lst) clean))

(else

(myRemove atm (cdr lst) (append clean (list (car lst))))

 )))

**7. A function (selectionSort list) that returns a simple list of integers in ascending order using a recursive *selection sort* algorithm. Hint: use your min function.**

* **Test selectionSort with lists (), (5), (6 10 23 12 2 9 18 1 0 15), (3 4 7 3 7 7 4 3 2 3 7)**

(define (selectionSort e)

(if (or (null? e) (<= (length e) 1)) e

(let loop ((left null) (right null)

(pivot (car e)) (rest (cdr e)))

(if (null? rest)

(append (append (selectionSort left) (list pivot)) (selectionSort right))

(if (<= (car rest) pivot)

(loop (append left (list (car rest))) right pivot (cdr rest))

(loop left (append right (list (car rest))) pivot (cdr rest)))))))

