3_1.rkt

```
#lang racket

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; HW: HW #11 CSC 600 Programming Language Design
; Author: Miguel Antonio Logarta
; Due: May 3, 2024
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;; Question 1a ;;;;;
; Anonymous function
(lambda (x) + x 1)

; Anonymous function used inside of a map function
(define numbers_1_a (list 1 2 3 4 5))
(map (lambda (x) (+ x 1)) numbers_1_a)

;;;;; Question 1b ;;;;;
(define (triple x)
   (* x 3))

; Function triple(x) is applied to every element in the list
(define numbers_1_b (list 1 2 3 4 5))
(map triple numbers_1_b)

;;;;; Question 1c ;;;;;
(define (double x)
   (* x 2))
(define (subtract_2 x)
   (- x 2))

; This is a list whose elements are functions
(define operations (list double subtract_2))

;;;;; Question 1d ;;;;;
; Comparing a function to see if they're the same
(define (tims_function x)
   (+ x 1))
(define (jerrys_function x)
   (- x 1))
(equal? tims_function tims_function) ; Outputs #t for true
(equal? tims_function jerrys_function) ; Outputs #f for false

; Comparing lists to see if they have the same values
(define numbers_1_d_first (list 1 2 3 4 5))
(define numbers_1_d_second (list 1 2 3 4 5))

(define numbers_1_d_third (list 2 4 6 8 10))

(equal? numbers_1_d_first numbers_1_d_second) ; Outputs #t for true
(equal? numbers_1_d_first numbers_1_d_third) ; Outputs #f for false

;;;;; Question 1e ;;;;;
```

```
(define (add_one x)
   (+ x 1))
(define (double_the_result x func)
   (* (func x) 2))

; double_the_result doubles the result of whatever function
; gets passed into it. The first arg is the value that is passed to
; whatever function is called
; (add_one 3) -> 4. (* 4 2) = 8
(double_the_result 3 add_one)

;;;;; Question 1f ;;;;;
(define (im_odd)
   (display "I'm an odd number\n"))
(define (im_even)
   (display "I'm an even number\n"))

; If x is an even number, return an even function, else return an odd function
(define (get_print_func x)
   (if (= (modulo x 2) 0)
     im_even
     im_odd))

; Get a function for odd numbers and execute it
(define (say_im_odd)
   ((get_print_func 3)))
(say_im_odd)

;;;;; Question 1g ;;;;;
; A function that reads your input
(define (get_user_input)
  (display "Enter some letters: ")
  (flush-output)
  (read-line))
(define (user_input)
   (get_user_input))

; Output what you entered
(displayln (user_input))

; A function that is read from a file
; example_file.rkt looks like this
   ; (provide import_this_function)
   ; (define (import_this_function)
   ;     (display "Hello everybody!\n"))
(require "example_file.rkt")
(import_this_function)

; A function that is displayed
(displayln import_this_function)

;;;;; Question 2 ;;;;;
; Computes the average of numbers passed to it
(define (average . rst)
  (/ (foldr + '0 rst) (length rst)))
```

```racket

; Squares every number using map, then adds them together using foldr
(define (average_of_squares . rst)
   (/ (foldr + '0 (map (lambda (x) (expt x 2)) rst)) (length rst)))

; Computes the standard deviation using average and average_of_squares
(define (sigma . rst)
   (sqrt (- (apply average_of_squares rst) (expt (apply average rst) 2))))

(sigma 1 2 3 2 1)
(sigma 1 3 1 3 1 3)
(sigma 1 3)
(sigma 1)

;;;;; Question 3a ;;;;;
; Prints out n stars in one line
(define (line n)
   (if (= n 0)
      ""
      (string-append "*" (line (- n 1)))))

(line 5)

;;;;; Question 3b ;;;;;
; Prints out n lines that contain m stars for each index in the list
(define (histogram lst)
   (for-each (lambda (x) (displayln (line x))) lst))

(histogram '(1 2 3 3 2 1))

;;;;; Question 4 ;;;;;
; (define (compute_max func x1 x2))

;;;;; Question 5a ;;;;;
; A * B = (Ax * Bx) + (Ay * By) + (Az * Bz) +...
(define (multiply_at_index vectorA vectorB index)
   (define vectorAIndex (vector-ref vectorA index))
   (define vectorBIndex (vector-ref vectorB index))
   (* vectorAIndex vectorBIndex))

(define (scalar-product_iterative vectorA vectorB)
   (if (= (vector-length vectorA) (vector-length vectorB))
      (do ([i 0 (+ i 1)])
         ((>= i (vector-length vectorA)))
         (displayln "an index"))
      (displayln "ERROR: Different sizes of vectors!")))

; Error different sizes of vectors is outputted
(scalar-product_iterative '#(1 2 3) '#(1 2 3 4 5))

(scalar-product_iterative '#(1 2 3) '#(2 1 1))


;;;;; Question 5b ;;;;;
;;;;; Question 6a ;;;;;
```

```
;;;;; Question 6b ;;;;;
```