Problem 1: Dot Product

The dot product of two vectors, \( u \) and \( v \), is \( \sum_i u_i \times v_i \).

Examples

Write two examples of the operation of dot-product.

Solution

\[
\begin{align*}
(\text{dot-product } (\text{make-vector } 5 1) (\text{make-vector } 5 1)) &= 5 \\
(\text{dot-product } (\text{make-vector } 5 1) (\text{make-vector } 5 2)) &= 10
\end{align*}
\]

Implementation

Write a function \textbf{dot-product} that computes the dot-product of two vectors.

\[
\begin{align*}
\text{(define (dot-product u v)} \\
; \quad \text{dot-product: (vectorof number) (vectorof number) -> number}
\end{align*}
\]

Solution

\[
\begin{align*}
\text{(define (dot-product-helper n v1 v2)} \\
; \quad \text{dot-product-helper: number vector-of-numbers vector-of-numbers -> +vector-of-numbers}
\end{align*}
\]

Test

Demonstrate the operation of your function on the examples you defined above.

Problem 2: Changeable Phonebook

Assume a variant of the phonebook in Homework 6, where instead of a list of structures, the phonebook is represented as a vector of structures as below.

A phone-book is a vector of length 100 where entries are either:

- \#f, or
- \text{(make-pb name number)}

\[(\text{define-struct pb (name number))}, \text{where name is a symbol and number is a number}\]
new-phonebook

Based on the definition above, create a new phonebook where all the entries are #f.

Solution

(define-struct pb (name number))
;; name : symbol
;; number : number

;; phonebook : vector of pb or #f
(define phonebook (make-vector 100 #f))
(define entries 0)

add-phonebook

Create a new function add-phonebook that inserts new phonebook entry - name and number - into the phonebook created above. If an entry already exists for a given name, do nothing.

(define (add-phonebook name number)
  ;; add-phonebook: symbol number -> (void)

  Solution

  ;; add-phonebook : symbol number -> void
  ;; adds the name and number to the phonebook
  ;; unless the name is already there.
  (define (add-phonebook name number)
    (cond
     [(is-in-phonebook? name entries) (void)]
     [(= (vector-length phonebook) entries) (void)]
     [else (begin (vector-set! phonebook entries (make-pb name number))
                    (set! entries (+ entries 1)))]))

(define (is-in-phonebook? name num)
  (cond
   [(zero? num) #f]
   [else (or (eq? name (pb-name (vector-ref phonebook (- num 1))))
             (is-in-phonebook? name (- num 1)))]))

update-phonebook

Implement a new function update-phonebook that takes a name and number and updates the associated phonebook entry if there is one, and returns #f, otherwise.
(define (update-phonebook name number)
 ;; add-phonebook: symbol number -> (void) or \#f

 Solution

 ;; update-phonebook : symbol number -> void
 (define (update-phonebook name number)
 ;; update-phonebook-helper : symbol number number -> void
 (update-phonebook-helper name number entries))

 (define (update-phonebook-helper name number n)
 (cond
  [(zero? n) (void)]
  [else (if (eq? (pb-name (vector-ref phonebook (- n 1))) name)
          (vector-set! phonebook (- n 1) (make-pb name number))
          (update-phonebook-helper name number (- n 1)))]))