MY FUNCTIONS

```
;; comparaison-number: List Number -> Number
;; comparaison-number h1 h2
;; Calculates the intersection similarity score between two
histograms represented as lists of numbers.
;; Input:
;;   - h1: A list representing the first histogram.
;;   - h2: A list representing the second histogram.
;; Output:
;;   - Returns the intersection similarity score between h1 and
h2.
(define (comparaison-number h1 h2)
  (cond
   ((null? h1) 0)
   ((> (car h1) (car h2))
              (+ (car h2) (comparaison-number (cdr h1) (cdr
h2))))
   ((< (car h1) (car h2))
              (+ (car h1) (comparaison-number (cdr h1) (cdr
h2))))
   (else (+ (car h1) (comparaison-number (cdr h1) (cdr h2))))
  )
)

;; hist-normaliseur: List -> List
;; hist-normaliseur h
;; Normalizes the given histogram by dividing each bin value by
the sum of all bin values.
;; Input:
;;   - h: A list representing the histogram.
;; Output:
;;   - Returns a list representing the normalized histogram.
(define (hist-normaliseur h)
  (let ((sum (apply + h)))
    (map (lambda (x) (/ x sum)) h)
   )
 )

;; similaritySearch: String String -> ListOfPairs
;; similaritySearch query-hist-filename directory-path
;; Given the filename of a query histogram file and the path to
a directory containing
;; other histogram files, this function computes the similarity
scores between the query
;; histogram and each histogram in the directory. It returns a
list of pairs, where each
```

```
;; pair consists of the filename of a histogram file and its
corresponding similarity score.
;; Input:
;;   - query-hist-filename: A string representing the filename
of the query histogram file.
;;   - directory-path: A string representing the path to the
directory containing other histogram files.
;; Output:
;;   - Returns a list of pairs, where each pair contains the
filename of a histogram file and its similarity score.
(define (similaritySearch query-hist-filename directory-path)
  (let* ((query-hist (hist-normaliseur (read-hist-file query-
hist-filename)))
         (scores (map (lambda (image-hist-filename)
                        (cons image-hist-filename
                              (exact->inexact (comparaison-
number query-hist (hist-normaliseur (read-hist-file (string-
append directory-path "/" image-hist-filename)))))))
                      (list-text-files-in-directory directory-
path)))
         (sorted-scores (take (sort scores (lambda (x y) (> (cdr
x) (cdr y)))) 5)))
    sorted-scores))
```

Reference :
Scheme YouTube playlist : https://youtube.com/playlist?list=PLgyU3jNA6VjRMB-LXXR9ZWcU3-GCzJPm0&si=KjEBzHUGPT1a_dKx