```
#| AMAR BESSEDIK
   PROJECT2: HUFFMAN CODING FOR DATA COMPRESSION IN LISP
   CSC 540
|#
;;;============================================================================
;;;---------------------------- MAKE-HUFFMAN-TREE.LISP  -----------------------
;;;============================================================================

;;; GENERATE THE HUFFMAN-TREE THAT CORRESPONDS TO A GIVEN MESSAGE (LIST OF SYMBOLS).
(defun make-huffman-tree (message)
  "generate the huffman-tree that corresponds to a given message (list of symbols)"
  (huffman-tree-builder (htree-sort (make-htrees message))))

(defun huffman-tree-builder (htrees)
  "merge, trim and sort sub-htrees"
  (cond ((= (length htrees) 1) (first htrees))
        (t  (huffman-tree-builder (htree-sort (trim-htrees (htree-merge (first htrees)
                                                                        (second htrees)) htree
s))))))


;;;===============  HELPER FUNCTIONS===========================

;;;GENERATE TREES
(defun make-htrees (message)
  "generate the sorted sub-htrees and put them in a list"
  (initialize-htrees (sort(freqlist message) #'<= :key #'second)))

;;;APPEND THE NEWLY CONSTRUCTED SUBHTREE FROM MERGE OPERATION
;;;ELIMINATE THE TWO
(defun trim-htrees (new-htree htrees)
  "append newly constructed sub-htree to the original list except the two operands"
  (append (list new-htree) (allbut 2 htrees)))

;;;ALL ELEMENTS OF A LIST EXCEPT THE FIRST N ELEMENTS USEFUL AFTER MERGING TWO HTREES,
;;;SINCE THOSE ELEMENTS SEPARATELY DO NOT BELONG TO THE BIG LIST OF TREES
(defun allbut (n htrees)
  "All but the n first element of a list"
  (if (and (listp htrees) (>= (length htrees) n))
      (cond ((= n 0) htrees)
            ((= n 1) (rest htrees))
            (t (allbut (1- n) (rest htrees))))))


;;;GIVEN A FREQUENCY LIST OF ALL SMBOLS, PUT EVERY MEMBER IN A LIST
;;;SO WE CAN EVERY SYMBOL AS A TREE OF ONE ELEMENT.
(defun initialize-htrees (frequencylist)
  "put each pair (symbol weight) in a list and the whole thing in a list"
  (if (not (null frequencylist))
      (cons (list(first frequencylist)) (initialize-htrees (rest frequencylist)))))
```