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

;;;RETURNS T IF WEIGHT OF HTREES1 IS LESS THEN THE WEIGHT OF HTREE2
;;;USEFUL FOR SORTING SUB-TREES IN INCREASING ORDER OF WEIGHTS
(defun htree-less (htree1 htree2)
  "returns t if weight of htrees1 is less then the weight of htree2"
  (< (second (first htree1)) (second (first htree2))))

;;;RETURNS THE LIST OF SYMBOLS STORED IN THE ROOT OF HTREE
(defun htree-symbols (htree)
  "returns the list of symbols stored in the root of htree"
  (first(first htree)))

;;;RETURNS THE WEIGHT OF HTREE
(defun htree-weight (htree)
  "Returns the weight of huffman tree"
  (if (numberp (second (first htree)))
      (second (first htree))
      (error "ERROR: WEIGHT PARAMETER IS NOT A NUMBER")))

;;;RETURNS THE ROOT OF A TREE
(defun root (htree)
  "Returns the root of huffman tree"
  (cond ((atom htree) htree)
        (t (first htree))))

;;;RETURNS A LIST OF SORTED HTREES BY THEIR INCREASING WEIGHT
(defun htree-sort (htrees)
  "Sorts huffman sub-trees from lesser weight to greater"
  (sort (copy-list htrees) #'htree-less))

;;;RETURNS THE RESULTED HUFFMAN TREE FROM MERGING HTREE1 & HTREE2
(defun htree-merge (htree1 htree2)
  "Merge two huffman sub-trees"
  (list (list (append (first(first htree1))
                      (first(first htree2)))
              (+ (htree-weight htree1) (htree-weight htree2)))
        htree1 htree2 ))

;;;RETURNS T IF A HTREE IS A LEAF
(defun leaf-p (htree)
  "Returns T if a node is a leaf"
  (null (rest htree)))

;;;RETURNS THE LEFT-SUBTREE
(defun left-subhtree (htree)
  "Returns T if a huffman sub-tree is a left branch, nil otherwise"
  (if (not (atom htree))
      (second htree)))

;;;RETURNS THE RIGHT-SUBTREE
(defun right-subhtree (htree)
  "Returns T if huffman sub-tree is a right branch, nil otherwise"
  (if (not (atom htree))
      (third htree)))
```