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

#| - THE FUNCTIONS IN THIS FILE USE BOTH THE FUNCTIONS IN
     'MAKE-HUFFMAN-TREE.LISP' AND 'ADT.LISP' TO DECODE A MESSAGE.
   - A MESSAGE IS A LIST OF BINARY CODE.
   - THE RETURNED VALUE WOULD BE A LIST SYMBOLS (ATOMS)|#

;;;=========================================================
#| - THE CODING SEQUENCES FOR EACH SYMBOL IS OF A VARIABLE LENGTH
   - GO THROUGH ALL BRANCHES LEADING TO A LEAF IN ORDER TO GET THE CODED SYMBOL IN QUESTION
   - RETURN THE SYMBOL AND GO BACK TO ROOT LEVEL TO DECODE THE NEXT SEQUENCE OF BITS.|#

;;;ALL THE HEAVY WORK OS DONE BY 'DECODER' FUNCTION
(defun decode (bits huffman-tree)
  "Decode all bits using decoder function"
  (decoder bits huffman-tree huffman-tree))

;;;TAKES A SEQUENCE OF BINARY DIGITS STARTING FROM 1ST BIT AND FOLLOW LEFT BRANCH IF BIT IS '0
'
;;;GO RIGHT IF BIT IS '1', ERROR OTHERWISE. WHEN A LEAF IS REACHED, THE CORRESPONDING SYMBOL I
S
;;;RETURNED
(defun decoder (bits current-branch huffman-tree)
  "decode sequence of bits starting from root;;
  When a leaf is reached, go back to root and start over gain"
  (if (not (null bits))
      (cond ((or (= (first bits) 0)
                 (= (first bits) 1))
             (cond ((leaf-p (next-decoding-branch (first bits) current-branch))
                    (cons (first(first(first (next-decoding-branch (first bits) current-branch
))))
                          (decoder (rest bits) huffman-tree huffman-tree)))
                   (t (decoder (rest bits) (next-decoding-branch (first bits) current-branch)
                               huffman-tree))))
            (t (error "ONLY BINARY NUMBER ARE ALLOWED FOR DECODING!!!")))))

;;;======================  HELPER FUNCTION ==================
;;;RETURNS THE LEFT SUB-TREE IF (BIT IS '0')
;;;RIGHT-SUB-TREE IF (BIT IS '1')
(defun next-decoding-branch (bit htree)
  "Returns Left branch if 0, Right branch if 1, error otherwise"
  (cond ((= bit 0) (left-subhtree htree))
        ((= bit 1) (right-subhtree htree))
        (t (error "ERROR !!! WRONG BIT"))))

;;;------- USEFUL FOR TESTING ONLY: NOT REQUIRED FOR GOOD FUNCTIONNING OF THE PROGRAM
;;;DECODE ONE CHARACTER AT A TIME
(defun decode-symbol (bit-sequence huffman-tree)
  "decode one character at a time"
  (if (zerop (first bit-sequence))
      (if (leaf-p (left-subhtree huffman-tree))
          (cons (first(first(first (left-subhtree huffman-tree)))) '())
          (decode-symbol (rest bit-sequence) (left-subhtree huffman-tree)))
      (if (leaf-p (right-subhtree huffman-tree))
          (cons (first(first(first (right-subhtree huffman-tree)))) '())
          (decode-symbol (rest bit-sequence) (right-subhtree huffman-tree)))))
```