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

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

#| ALL THE HEAVY WORK IS DONE BY "ENCODE-SYMBOL" FUNCTION. THE ROLE OF "ENCODE"
   FUNCTION IS GOING THROUGH THE WHOLE LIST OF SYMBOLS AND GENERATE A LIST OF BINARY
   CODE. NOTE: THERE ARE NO SEPARATORS BETWEEN BINARY OF 1 CHARACTER & ANOTHER |#
(defun encode (message huffman-tree) ;; PROVIDE THE APPROPRIATE MESSAGE LIST
               (cond ((null message) '()) ;; IF MESSAGE ENPTY, RETURN EMPTY LIST
                     ;;ENCODE 1ST SYMBOL AND APPEND IT TO THE REST TO ENCODE
                     (t (append (encode-symbol (first message) huffman-tree)
                                (encode (rest message) huffman-tree)))))


#|- ENCODE ONE SYMBOL AT A TIME. ---> STARTING FROM THE ROOT OF HUFFMAN TREE:
 - ADD 0 WHEN EVER WE GO LEFT. - ADD 1 WHEN EVER WE GO RIGHT.
 - THE CONSTRUCTED SEQUENCE OF BITS WILL BE THE HUFFMAN CODE FOR THE SYMBOL IN QUESTION.|#

(defun encode-symbol (symbol huffman-tree);; For a given symbol
  (if (leaf-p (next-branch symbol huffman-tree))
      ;;; IF THE NEXT ENCODING BRANCH IS A 'LEAF'
      (if (equal (next-encoding-branch symbol huffman-tree) (left-subhtree huffman-tree))
          (cons 0 '()) ;; CONSTRUCT A '0' IF WE ARE ON THE LEFT HAND SIDE OF TREE
          (cons 1 '()));; CONSTRUCT A '1' IF WE ARE ON THE RIGHT HAND SIDE OF TREE
      ;;; IF THE NEXT ENCODING BRANCH IS 'NOT A LEAF'
      (if (equal (next-encoding-branch symbol huffman-tree) (left-subhtree huffman-tree))
          ;;;CONTINUE ENCODING FLLOWING THE LEFT ENCODING BRANCH
          (cons 0 (encode-symbol symbol (left-subhtree huffman-tree)))
          ;;;CONTINUE ENCODING FLLOWING THE RIGHT ENCODING BRANCH
          (cons 1 (encode-symbol symbol (right-subhtree huffman-tree))))))

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

;;; GIVEN A SYMBOL, RETURNS THE SUB-HTREE TO WHICH IT BELONGS
;;; ERROR IF THE SYMBOL IS NEITHER IN LEFT NOR IN RIGHT SUB-HTREES

(defun next-encoding-branch (symbol htree)
  (if (not (null htree))
      (cond ((member-p symbol (htree-symbols (left-subhtree htree)))
             (left-subhtree htree))
            ((member-p symbol (htree-symbols (right-subhtree htree)))
             (right-subhtree htree))
            (t (write ("No such symbol in Tree"))))))

;;; PREDICATE TO TEST IF A SYMBOL BELONGS TO A SET OF SYMBOLS
(defun member-p (s symbols)
  (if (null symbols) nil
      (if (eql s (first symbols)) t
          (member-p s (rest symbols)))))
```