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

;;;  GENERATE THE FREQUENCY LIST FROM A MESSAGE

;;;GENERATE FREQUENCY LIST
;;;ALL THE HEAVY WORK IS DONE BY 'build-frequency-list' function
(defun freqlist (message)
  "Returns the frequency list out of symbols of a message"
  (build-frequency-list message nil))

;;;BUILD A FREQUENCY LIST OUT OF A MESSAGE GRADUALLY EACH TIME A WORD IS ENCOUNTRED
(defun build-frequency-list (message frequencies)
  "go through a message & build the frequency list step by step using 'update' function"
  (cond ((endp message) frequencies)
        (t (build-frequency-list (rest message)(update (first message) frequencies)))))

;;;UPDATE A FREQUENCY OF A WORD BY INCREMETING ITS WEIGHT BY 1 EACH TIME IT IS ENCOUNTRED
;;;WHILE A MESSAGE IS BEING READ
(defun update (word frequencies)
  "Update a frequency of a word by incrementing its weight by 1"
  (cond ((endp frequencies) (list(list(list word) 1)))
        ((fequal word (first(first frequencies)))
         (cons (incpair(first frequencies))(rest frequencies)))
        ( t (cons (first frequencies)(update word (rest frequencies))))))

;;;WHILE GOING THROUGH A MESSAGE, TEST IF A WORD IS A PAIR MEMBER
(defun fequal (word pair) (equal word (first pair)))

;;;PUT A WORD IN A LIST ALONG WITH ITS INCREMENTED WEIGHT
(defun incpair (pair) (list (first pair)(1+ (second pair))))
```