File created: 18-Nov-88 10:52:55 **{ERINYES}<LISPUSERS>MEDLEY>WORDNERD.;1**

changes to: (FNS WordNerd.DefaultVennSearch WordNerd.DefaultWeightedSearch HashfileNerd.ExpandKeyPattern HashfileNerd.MapKeys HashfileNerd.Create)
(VARS WORDNERDCOMS)
(MACROS WordNerd.ExpandKeyPattern)
(RECORDS WNKEYSETINFO WNKEYINFO)

previous date: 11-Nov-88 17:23:39 {QV}<DICTSERVER>LISP>WORDNERD.;24

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ WORDNERDCOMS
       ((* * The following macros are the interface to three different search techniques using a WordNerd. The
          default implementations are given below.)
        (MACROS WordNerd.Open WordNerd.Close WordNerd.AddAssociation WordNerd.MapKeys WordNerd.ExpandKeyPattern
                WordNerd.VennSearch WordNerd.RelevanceSearch WordNerd.WeightedSearch)
        (FNS WordNerd.AddEntry WordNerd.AddDictionary WordNerd.AddStopWords WordNerd.SortByFrequency)
        (FNS WordNerd.DefaultVennSearch)
        (FNS WordNerd.DefaultWeightedSearch AddWeightsToArray FindTopElements AddToPriorityList)
        (FNS WordNerd.DefaultRelevanceSearch MergeKeywords)
        (FNS WORDNERD.PARSEINPUT)
        (RECORDS WNKEYINFO)
        (* * SimpleNerd is an in-core version of the WordNerd.)
        (COMS (FNS SimpleNerd.Create SimpleNerd.AddAssociation SimpleNerd.MapKeys SimpleNerd.GetEntry
                   SimpleNerd.MaxEntry SimpleNerd.GetHeader SimpleNerd.ParseDictEntry SimpleNerd.Test))
        (* * HashfileNerd stores its data structures in an InterLisp hashfile.)
        (COMS (FNS HashfileNerd.Create HashfileNerd.Test HashfileNerd.Open HashfileNerd.Close HashfileNerd.Write
                   SIMPLETYPE HashfileNerd.AddAssociation HashfileNerd.GetEntry HashfileNerd.ExpandKeyPattern
                   HashfileNerd.MapKeys)
              (FNS BIGHASH BIGGETHASH BIGHASHSIZE BIGMAPHASH BIGPUTHASH)
              (MACROS BIGHASHP)
              (RECORDS BIGHASH))
        (* * the following should be merged into ANALYZER eventually.)
        (COMS (FNS FileDict.Create FileDict.AddFiles FileDict.PrintEntry FileDict.Write FileDict.Lookup
                   FileDict.MapEntries FETCHSTRINGFROMFILE)
              (FNS SimpleAnalyzer.Create SimpleAnalyzer.Lookup)
              (FNS SimpleDict.Create SimpleDict.Open SimpleDict.Close SimpleDict.Write))
        (VARS ENGLISHSTOPWORDS)))
```

(* * The following macros are the interface to three different search techniques using a WordNerd.
The default implementations are given below.)

```
(DECLARE%: EVAL@COMPILE

(PUTPROPS WordNerd.Open MACRO ((WORDNERD)
                               (APPLY* (OR (InvertedDict.Prop WORDNERD 'OPENFN)
                                           (FUNCTION NILL))
                                       WORDNERD)))

(PUTPROPS WordNerd.Close MACRO ((WORDNERD)
                                (APPLY* (OR (InvertedDict.Prop WORDNERD 'CLOSEFN)
                                            (FUNCTION NILL))
                                        WORDNERD)))

(PUTPROPS WordNerd.AddAssociation MACRO ((WORDNERD HEADER KEY)
                                         (APPLY* (InvertedDict.Prop WORDNERD 'ADDASSOCIATIONFN)
                                                 WORDNERD HEADER KEY)))

(PUTPROPS WordNerd.MapKeys MACRO ((WORDNERD MAPFN)
                                  (APPLY* (OR (InvertedDict.Prop WORDNERD 'MAPKEYSFN)
                                              (FUNCTION NILL))
                                          WORDNERD MAPFN)))

(PUTPROPS WordNerd.ExpandKeyPattern MACRO ((WORDNERD KEYPATTERN)
                                           (APPLY* (OR (InvertedDict.Prop WORDNERD 'EXPANDKEYPATTERNFN)
                                                       (FUNCTION NILL))
                                                   WORDNERD KEYPATTERN)))

(PUTPROPS WordNerd.VennSearch MACRO ((WORDNERD SYNONYMCLASSES MINKEYWORDS MINWORD MAXWORD DONTCONVERT)
                                     (APPLY* (InvertedDict.Prop WORDNERD 'VENNSEARCHFN)
                                             WORDNERD SYNONYMCLASSES MINKEYWORDS MINWORD MAXWORD DONTCONVERT)))

(PUTPROPS WordNerd.RelevanceSearch MACRO ((WORDNERD HEADERS KEYSTOIGNORE MINWORD MAXWORD)
                                          (APPLY* (InvertedDict.Prop WORDNERD 'RELEVANCESEARCHFN)
                                                  WORDNERD HEADERS KEYSTOIGNORE MINWORD MAXWORD)))

(PUTPROPS WordNerd.WeightedSearch MACRO ((WORDNERD WEIGHTEDKEYS MINWORD MAXWORD USEFREQWEIGHTS)
```

```
                                        (APPLY* (InvertedDict.Prop WORDNERD 'WEIGHTEDSEARCHFN)
                                               WORDNERD WEIGHTEDKEYS MINWORD MAXWORD USEFREQWEIGHTS)))
)

(DEFINEQ
```

(WordNerd.AddEntry

```
  [LAMBDA (WORDNERD HEADER ENTRY ANALYZER)                    ; Edited 14-Sep-88 09:25 by jtm:
    (LET (ADDASSOCFN)
         [COND
            ((NULL ANALYZER)
             (SETQ ANALYZER (InvertedDict.Prop WORDNERD 'ANALYZER]
         (SETQ ADDASSOCFN (InvertedDict.Prop WORDNERD 'ADDASSOCIATIONFN))
         (Analyzer.Analyze ANALYZER ENTRY NIL NIL (FUNCTION (LAMBDA (ANALYZER STREAM START LENGTH VAL)
                                                              (APPLY* ADDASSOCFN WORDNERD HEADER
                                                                     (OR VAL (STREAM.FETCHSTRING STREAM START
                                                                                  LENGTH NIL T)))
                                                             NIL])
```

(WordNerd.AddDictionary

```
  [LAMBDA (WORDNERD DICTIONARY ANALYZER)                      ; Edited 14-Sep-88 10:11 by jtm:
    [COND
       ((NULL ANALYZER)
        (SETQ ANALYZER (InvertedDict.Prop WORDNERD 'ANALYZER]
    (Dict.MapEntries DICTIONARY (COND
                                  [(EQ (FUNCTION SimpleDict.MapEntries)
                                       (fetch (Dict mapFn) of DICTIONARY))
                                                              (* this is a hack until we fix SimpleDict.MapEntries)
                                   (FUNCTION (LAMBDA (DICT HEADER ENTRY)
                                               (WordNerd.AddEntry WORDNERD (CONCATLIST HEADER)
                                                      ENTRY ANALYZER]
                                  (T (FUNCTION (LAMBDA (DICT HEADER ENTRY)
                                                 (WordNerd.AddEntry WORDNERD HEADER ENTRY ANALYZER])
```

(WordNerd.AddStopWords

```
  [LAMBDA (WORDNERD STOPWORDS)                                ; Edited 15-Sep-88 17:12 by jtm:
    (for WORD inside STOPWORDS do (WordNerd.AddAssociation WORDNERD :STOPWORD WORD))
    WORDNERD])
```

(WordNerd.SortByFrequency

```
  [LAMBDA (WORDNERD MINCOUNT)                                 ; Edited 25-Oct-88 13:34 by jtm:
    (LET (ENTRIES)
         [WordNerd.MapKeys WORDNERD (FUNCTION (LAMBDA (NERD KEYWORD KEYID ASSOCS)
                                                (DECLARE (SPECVARS MINCOUNT))
                                                (LET (FREQ)
                                                     [SETQ FREQ (COND
                                                                   ((LISTP ASSOCS)
                                                                    (LENGTH ASSOCS]
                                                     (COND
                                                        ((AND FREQ (OR (NULL MINCOUNT)
                                                                       (ILEQ MINCOUNT FREQ)))
                                                         (push ENTRIES (LIST FREQ KEYWORD]
         [SETQ ENTRIES (SORT ENTRIES (FUNCTION (LAMBDA (A B)
                                                 (IGREATERP (CAR A)
                                                        (CAR B]
         ENTRIES])

)

(DEFINEQ
```

(WordNerd.DefaultVennSearch

```
  [LAMBDA (wordNerd synonymClasses minKeywords minWord maxWord dontConvert)
                                                              ; Edited 17-Nov-88 17:14 by jtm:
    (LET (analyzer keys venn first keySet keySetWeight keySetInfo notFound priorSet headerIndex keyIndex
                indexFile GetKeyIDFn GetHeaderFn GetBufferFn GetEntryFn ExpandKeyPatternFn (n 0))
         ;; initialize the word nerd
         [COND
            ((AND wordNerd (LITATOM wordNerd))
             (SETQ wordNerd (InvertedDictFromName wordNerd]
         [COND
            ((NULL wordNerd)
             (SETQ wordNerd (CAR InvertedDict.List]
         (WordNerd.Open wordNerd)
         ;; canonicalize the user input
         [COND
            ((STRINGP synonymClasses)
             (SETQ synonymClasses (WORDNERD.PARSEINPUT wordNerd synonymClasses]
         ;; cache the object-oriented fields and functions in local variables.
         (SETQ headerIndex (fetch (INVERTEDDICT HEADERINDEX) of wordNerd))
         (SETQ keyIndex (fetch (INVERTEDDICT KEYINDEX) of wordNerd))
```

```
(SETQ indexFile (fetch (INVERTEDDICT INDEXFILE) of wordNerd))
(SETQ GetKeyIDFn (InvertedDict.Prop wordNerd 'GETKEYIDFN))
(SETQ GetEntryFn (InvertedDict.Prop wordNerd 'GETENTRYFN))
(SETQ GetHeaderFn (InvertedDict.Prop wordNerd 'GETHEADERFN))
(SETQ GetBufferFn (InvertedDict.Prop wordNerd 'GETBUFFERFN))
(SETQ ExpandKeyPatternFn (InvertedDict.Prop wordNerd 'EXPANDKEYPATTERNFN))
```

;; synonymClasses is a list of lists of words, where each sub-list represents a class of synonyms

```
[for synonymClass wordList inside synonymClasses
   do (SETQ wordList NIL)
```

;; build wordlist, a concatenation of the entries in the synonym class

```
      [for pattern inside synonymClass
         do (for word word# entry inside (OR (AND ExpandKeyPatternFn (STRPOS '* pattern)
                                                  (APPLY* ExpandKeyPatternFn wordNerd pattern))
                                             pattern)
               do (COND
                     [[SETQ word# (COND
                                     ((OR (NUMBERP word)
                                          (NULL GetKeyIDFn))
                                      word)
                                     (T (APPLY* GetKeyIDFn wordNerd word keyIndex]
                      (SETQ entry (APPLY* GetEntryFn wordNerd word# indexFile))
                      (COND
                         ((OR (EQ entry :STOPWORD)
                              (EQ -1 (CAR entry)))
                          (SETQ entry NIL)))
                      (SETQ wordList (COND
                                        ((LISTP synonymClass)
                                                          ; NCONC would destructively modify the wordNerd's entry.
                                         (APPEND entry wordList))
                                        (T entry]
                      (T (push notFound (create WNKEYINFO
                                                WNKEY _ (LIST (CONCAT word "?"]
      (SETQ wordList (SORT wordList (FUNCTION ILESSP)))
                                                          ; SORT may modify the entry, but shouldn't be a problem.
```

;; add the word list to the list of key associations

```
      (AND wordList (push keys (create WNKEYINFO
                                       WNKEY _ (COND
                                                  [(LISTP synonymClass)
                                                   (COND
                                                      ((CDR synonymClass)
                                                       (CONCAT (CAR synonymClass)
                                                               "+"))
                                                      (T (CAR synonymClass]
                                                  (T synonymClass))
                                       WNKEYWEIGHT _ (IQUOTIENT 10000 (LENGTH wordList))
                                       WNKEYDATA _ wordList]
(SETQ keys (DREVERSE keys))
```

;; determine the minimum number of keys for a word to be included in the result

```
(OR minKeywords (SETQ minKeywords 2))
(COND
   ((ILEQ minKeywords 0)
    (SETQ minKeywords (IPLUS (LENGTH keys)
                             minKeywords)))
   ((ILESSP (LENGTH keys)
            minKeywords)
    (SETQ minKeywords 1)))
```

;; set up minWord and maxWord

```
(COND
   ((NULL minWord)
    (SETQ minWord 0))
   ((EQ minWord 1)                                    ; minWord = 0 allows notFound to be returned.
    (SETQ minWord 0)))
(COND
   ((OR (NULL maxWord)
        (EQ maxWord 0))
    (SETQ maxWord 65000)))
```

;; now skim the classes off of the top of the lists in alphabetical order, putting them in a Venn diagram

```
[do                                                   ; find the lowest numbered entry in the word lists
    (SETQ first NIL)
    [for keyInfo myFirst in keys do (SETQ myFirst (CAR (fetch WNKEYDATA of keyInfo)))
                                    (COND
                                       ((OR (NULL first)
                                            (AND myFirst (ILESSP myFirst first)))
                                        (SETQ first myFirst]
    (COND
       ((NULL first)                                  ; all of the word lists are empty
        (RETURN))
       (T ;; make a list of all of the classes that have 'first' in their word list.  Remove 'first' from the word lists
          (SETQ keySet NIL)
          (SETQ keySetWeight 0)
```

```
                    [for keyInfo myFirst myKey in keys
                       do (SETQ myKey (fetch WNKEY of keyInfo))
                          (SETQ myFirst (CAR (fetch WNKEYDATA of keyInfo)))
                          (COND
                             ((AND myFirst (EQP first myFirst))
                              (COND
                                 ((NULL keySet)
                                  (push keySet myKey))
                                 (T (NCONC1 keySet myKey)))
                              (add keySetWeight (fetch WNKEYWEIGHT of keyInfo))
                              ;; eliminate multiple entries
                              (while (AND (SETQ myFirst (CAR (fetch WNKEYDATA of keyInfo)))
                                          (EQP myFirst first))
                                 do (pop (fetch WNKEYDATA of keyInfo]
                                                                  ; put 'first' in the appropriate venn diagram entry
                    (COND
                       ((IGEQ (LENGTH keySet)
                              minKeywords)                        ; skip single classes if we were given multiple key classes
                        (COND
                           ([NOT (SETQ keySetInfo (for set in venn thereis (EQUAL keySet (fetch WNKEY of set]
                            (SETQ keySetInfo (create WNKEYINFO
                                                     WNKEY _ keySet
                                                     WNKEYWEIGHT _ keySetWeight))
                            (push venn keySetInfo)))
                        (push (fetch WNKEYDATA of keySetInfo)
                              first]
```

;;; sort the venn diagram so that the classes that are in the most overlaps come first

```
          [SORT venn (FUNCTION (LAMBDA (A B)
                                 (LET [(ALEN (LENGTH (fetch WNKEY of A)))
                                       (BLEN (LENGTH (fetch WNKEY of B]
                                      (COND
                                         ((EQ ALEN BLEN)
                                          (IGREATERP (fetch WNKEYWEIGHT of A)
                                                     (fetch WNKEYWEIGHT of B)))
                                         (T (IGREATERP ALEN BLEN]
```

;;; extract words in the range '[minWord..maxWord].' Convert numbers into lemmas

```
          (SETQ priorSet NIL)
          (SETQ venn (NCONC notFound venn))
          [for tail keySet keySetLength overflow i (buffer _ (AND GetBufferFn (APPLY* GetBufferFn wordNerd
                                                                                      headerIndex)))
             on venn do (SETQ overflow NIL)
                        (SETQ keySet (CAR tail))
                        (SETQ keySetLength (LENGTH (fetch WNKEYDATA of keySet)))
                        (COND
                           [(IGEQ n maxWord)                      ; set is above maximum, remove from venn diagram
                            (COND
                               [(EQ minWord 0)
                                (replace WNKEYDATA of keySet with (LIST (CONCAT keySetLength " entries."]
                               (T (COND
                                     (priorSet (RPLACD priorSet (CDR tail)))
                                     (T (SETQ venn (CDR tail]
                           [(ILESSP (IPLUS n keySetLength)
                                    minWord)                      ; set is below minimum, remove from venn diagram
                            (add n keySetLength)
                            (COND
                               ((NULL (CDR tail))
                                ;; if all of the sets are below minimum, leave the header for the last one so that the user knows what is going
                                ;; on.
                                (push (fetch WNKEY of keySet)
                                      ". . .")
                                (replace WNKEYDATA of keySet with (LIST "no more words.")))
                               (T (SETQ venn (CDR tail]
                           (T                                     ; we want to include at least part of this set
                              (SETQ priorSet tail)
                              (replace WNKEYDATA of keySet with (DREVERSE (fetch WNKEYDATA of keySet)))
                              [for keyTail on (fetch WNKEYDATA of keySet)
                                 do (add n 1)
                                    (COND
                                       ((AND (IGEQ n minWord)
                                             (NOT dontConvert))   ; convert the number into a word
                                        (RPLACA keyTail (CONCAT (APPLY* GetHeaderFn wordNerd (CAR keyTail)
                                                                        headerIndex buffer)))
                                                                  ; CONCAT will copy the string out of the buffer.
                                        (BLOCK)))
                                    (COND
                                       ((EQ n minWord)            ; remove the numbers before this one
                                        (replace WNKEYDATA of keySet with keyTail)
                                        (push (fetch WNKEY of keySet)
                                              ". . ."))
                                       ((AND (IGEQ n maxWord)
                                             (CDR keyTail))       ; remove the numbers after this one
```

```
                                                        ; add overflow (LENGTH (CDR lemma))
                                    (SETQ overflow (LENGTH (CDR keyTail)))
                                    (RPLACD keyTail NIL)
                                    (RETURN]
                                (replace WNKEYDATA of keySet with (SORT (fetch WNKEYDATA of keySet)
                                                                        (FUNCTION UALPHORDER)))
                                [COND
                                   (overflow (NCONC1 (fetch WNKEYDATA of keySet)
                                                     (CONCAT ". . .+" overflow " more."]
                                ;; finally, remove the WNKEYSETWEIGHT field
                                (RPLACD keySet (CDDR keySet]
          ;; COND ((NEQ overflow 0) (* append the overflow information) (NCONC1 (CADAR (LAST venn)) (CONCAT '. . .+' overflow ' more.')))
          venn])

)

(DEFINEQ
```

(WordNerd.DefaultWeightedSearch

```
  [LAMBDA (wordNerd weightedKeys minWord maxWord useFreqWeights) ; Edited 18-Nov-88 10:09 by jtm:
```

;;; performs a weighted search of wordNerd using the keys and weights in weightedKeys. minWord and maxWord gives the range of the result to be
;;; returned. useFreqWeights indicates that frequency should be taken into account. weightedKeys is either a string of keys from the user or a list of
;;; keys or a list of key-weight pairs (CAR, CADR).

```
    (LET (headerIndex keyIndex indexFile array arrayMax entryCount priorityList weights minimumWeight
                shiftFactor maxKeys wordList keysLeft GetKeyFn GetEntryFn GetFreqFn GetHeaderFn GetBufferFn
                MaxHeaderFn)
```

;;; the user may specify a database by name.

```
          [COND
             ((AND wordNerd (LITATOM wordNerd))
              (SETQ wordNerd (InvertedDictFromName wordNerd]
```

;;; The INVERTEDDICT data structure simulates an object-oriented approach. Fetch all of the values once at the beginning to save the cost of multiple
;;; fetches.

```
          (WordNerd.Open wordNerd)
          (OR minWord (SETQ minWord 0))
          (OR maxWord (SETQ maxWord 50))
          (SETQ minimumWeight (OR (InvertedDict.Prop wordNerd 'MINIMUMWEIGHT)
                                  0))
          (SETQ shiftFactor (OR (InvertedDict.Prop wordNerd 'SHIFTFACTOR)
                                0))
          (SETQ maxKeys (OR (InvertedDict.Prop wordNerd 'MAXWEIGHTEDKEYS)
                            15))
          (SETQ headerIndex (fetch (INVERTEDDICT HEADERINDEX) of wordNerd))
          (SETQ keyIndex (fetch (INVERTEDDICT KEYINDEX) of wordNerd))
          (SETQ indexFile (fetch (INVERTEDDICT INDEXFILE) of wordNerd))
          (SETQ GetKeyFn (InvertedDict.Prop wordNerd 'GETKEYFN))
          (SETQ GetEntryFn (InvertedDict.Prop wordNerd 'GETENTRYFN))
          (SETQ GetFreqFn (InvertedDict.Prop wordNerd 'GETFREQFN))
          (SETQ MaxHeaderFn (InvertedDict.Prop wordNerd 'MAXHEADERIDFN))
          (SETQ GetHeaderFn (InvertedDict.Prop wordNerd 'GETHEADERFN))
          (SETQ GetBufferFn (InvertedDict.Prop wordNerd 'GETBUFFERFN))
          (SETQ entryCount (OR (APPLY* MaxHeaderFn wordNerd headerIndex)
                               10000))
          (SETQ arrayMax (ADD1 (LRSH entryCount 8)))
```

;;; Cache the array scratch pad on invertedDict.

```
          (SETQ array (InvertedDict.Prop wordNerd 'Array))
          (COND
             ((AND array (IGREATERP arrayMax (ARRAYSIZE array)))  ; the data has grown since we last saw it.
              (SETQ array NIL)))
          [COND
             ((NULL array)
              (InvertedDict.Prop wordNerd 'Array (SETQ array (ARRAY (ADD1 arrayMax)
                                                                    NIL NIL 0]
          (for I from 0 to arrayMax do (SETA array I NIL))
```

;;; parse the user's input.

```
          (AND (STRINGP weightedKeys)
               (SETQ weightedKeys (WORDNERD.PARSEINPUT wordNerd weightedKeys T)))
```

;;; process the keys.

```
          [for word word# weight length freq factor inside weightedKeys
             do (SETQ wordList NIL)
                (COND
                   ((LISTP word)
                    (SETQ word (CAR word))
                    (SETQ factor (CADR word)))
```

```
                (T (SETQ factor 1)))
             (COND
                ([SETQ word# (COND
                                ((OR (NULL GetKeyFn)
                                     (NUMBERP word))
                                 word)
                                (T (APPLY* GetKeyFn wordNerd word keyIndex]
                 [SETQ freq (COND
                               ((LISTP word#)
                                (LENGTH word#))
                               ((NULL GetFreqFn)
                                (SETQ word# (APPLY* GetEntryFn wordNerd word# indexFile))
                                (LENGTH word#))
                               (T (APPLY* GetFreqFn wordNerd word# indexFile]
                 (COND
                    ((NEQ freq 0)
                     (SETQ weight (COND
                                     [useFreqWeights (ITIMES factor (IMAX 1 (LRSH (IQUOTIENT entryCount freq)
                                                                                  shiftFactor]
                                     (T factor)))
                     (COND
                        ((IGEQ (ABS weight)
                               minimumWeight)
                         (push weights (LIST word# weight word wordList]
```

;;; sort weights from greatest to least. We may not have to process all of the keys, so do the most significant ones first.

```
          [SETQ weights (SORT weights (FUNCTION (LAMBDA (A B)
                                                  (IGREATERP (CADR A)
                                                         (CADR B]
          [SETQ keysLeft (COND
                            (useFreqWeights maxKeys)
                            (T (LENGTH weightedKeys]
          [for tail weightedKey priorTail word weight singleWeight ignoreSingletons (%#entries _ 0) on weights
             do (BLOCK)
                (SETQ weightedKey (CAR tail))
                (SETQ weight (CADR weightedKey))
                (SETQ word (CADDR weightedKey))
                [SETQ wordList (COND
                                  ((LISTP (CAR weightedKey))
                                   (CAR weightedKey))
                                  (T (APPLY* GetEntryFn wordNerd (CAR weightedKey)
                                             indexFile]
                (RPLACA weightedKey word)
                (RPLACD (CDR weightedKey)
                        NIL)
                (COND
                   [(CDR wordList)                                  ; ignore lists with less than 2 and more than 2000 entries.
                    (add keysLeft -1)
                    (SETQ priorTail tail)
                    [COND
                       ((AND singleWeight (NOT ignoreSingletons))
                        (SETQ ignoreSingletons (IGEQ singleWeight
                                                     (for remaining in tail as I from 1 to keysLeft
                                                        sum (CADR remaining]
                    (add %#entries (AddWeightsToArray array wordList weight word ignoreSingletons))
                    [COND
                       ((AND maxWord (IGEQ %#entries maxWord))      ; keep track of the maximum weight of any key that could satisfy
                                                                    ; the query all by itself.
                        (COND
                           ((OR (NULL singleWeight)
                                (IGREATERP weight singleWeight))
                            (SETQ singleWeight weight]
                    (COND
                       ((EQ keysLeft 0)
                        (RPLACD tail NIL)
                        (RETURN]
                   (priorTail (RPLACD priorTail (CDR tail)))
                   (T (SETQ weights (CDR tail]
          (SETQ priorityList (CDR (FindTopElements array maxWord arrayMax)))
          (AND minWord (IGREATERP minWord 0)
               (SETQ priorityList (NTH priorityList minWord)))
          [SETQ priorityList (for lemma (buffer _ (AND GetBufferFn (APPLY* GetBufferFn wordNerd headerIndex)))
                                in priorityList collect             ; CONCAT will copy the string out of the buffer.
                                                  (CONS (CONCAT (APPLY* GetHeaderFn wordNerd (CADR lemma)
                                                                        headerIndex buffer))
                                                        (LIST (CAR lemma)
                                                              (DREVERSE (CDDDR lemma]
          (LIST weights priorityList])
```

(AddWeightsToArray

```
  [LAMBDA (array wordList weight word ignoreSingletons)          (* jtm%: "17-Nov-87 14:49")
    (for header index val elt (%#newEntries _ 0) in wordList
       do (SETQ index (LRSH header 8))
          (SETQ elt (ELT array index))
          (COND
```

```
            ((AND [NULL (SETQ val (for I in elt thereis (EQP header (CAR I]
                  (NOT ignoreSingletons))
             (SETQ val (LIST header 0))
             (COND
                (elt (ATTACH val elt))
                (T (push (ELT array index)
                         val)))
             (add %#newEntries 1)))
          (COND
             (val (push (CDDR val)
                        word)
                  (add (CADR val)
                       weight)))
       finally (RETURN %#newEntries])
```

(FindTopElements

```
  [LAMBDA (array maxWord arrayMax)                          (* jtm%: " 2-Aug-88 10:37")
    (LET (priorityList)
         (for I from 0 to arrayMax do (for arrayVal in (ELT array I) do
                                                          (* RPLACA (CDR arrayVal) (ITIMES
                                                          (CADR arrayVal) (IMIN 5 (LENGTH
                                                          (CDDR arrayVal)))))
                                                               (SETQ priorityList (AddToPriorityList
                                                                                   priorityList arrayVal
                                                                                   (CADR arrayVal)
                                                                                   maxWord)))
                                      (SETA array I NIL))
         priorityList])
```

(AddToPriorityList

```
  [LAMBDA (priorityList I VAL MAX)                          (* jtm%: " 6-Nov-87 15:12")
    (LET (inserted)
         [COND
            [(NULL priorityList)                            (* include a count at the beginning.)
             (SETQ priorityList (CONS (CONS 0 1)
                                      (LIST (CONS VAL I]
            ((AND MAX (ILEQ MAX (CDAR priorityList))
                  (ILEQ VAL (CAAR priorityList)))           (* its off the bottom)
             NIL)
            (T (for tail nextToLast last on priorityList as N from 0
                  do (COND
                        ((OR (NULL (CDR tail))
                             (IGREATERP VAL (CAADR tail)))
                         (COND
                            ((EQ N MAX)
                             NIL)
                            ((AND [SETQ nextToLast (AND MAX (NTH tail (IDIFFERENCE MAX N]
                                  (SETQ last (CDR nextToLast)))  (* re-use the nextToLast cell.)
                             (RPLACA (CAR priorityList)
                                    (CAAR nextToLast))
                             (RPLACD nextToLast NIL)            (* remove last from the list.)
                             (RPLNODE (CAR last)
                                    VAL I)                      (* update its values.)
                             (RPLACD last (CDR tail))           (* splice it into the list.)
                             (RPLACD tail last))
                            (T                                  (* (AND MAX (RPLACD (NTH tail
                                                                (IDIFFERENCE (ADD1 MAX) N)) NIL)))
                               (RPLACD tail (CONS (CONS VAL I)
                                                  (CDR tail)))
                               (add (CDAR priorityList)
                                    1)))
                         (RETURN]
         priorityList])

)

(DEFINEQ
```

(WordNerd.DefaultRelevanceSearch

```
  [LAMBDA (wordNerd posWords negKeys minWord maxWord)       (* jtm%: " 2-Aug-88 10:46")

          (* * extract keywords from the sample words given (posWords) and do a weighted search.)

    (LET (posKeys dictionary analyzer GetEntryTokensFn)
         [COND
            ((AND wordNerd (LITATOM wordNerd))
             (SETQ wordNerd (InvertedDictFromName wordNerd]
         [COND
            ((NULL wordNerd)
             (SETQ wordNerd (CAR InvertedDict.List]
         [COND
            ((STRINGP posWords)
             (SETQ posWords (PARSEBYCOLONS posWords]
         (SETQ analyzer (InvertedDict.Prop wordNerd 'ANALYZER))
         (SETQ dictionary (InvertedDict.Prop wordNerd 'DICTIONARY))
```

```
        (SETQ GetEntryTokensFn (InvertedDict.Prop wordNerd 'GETENTRYTOKENSFN))

         (* * GetEntryTokensFn is in the wordNerd rather than its dictionary because there may be more than one wordNerd for a
         particular dictionary (as in the WordNerd and EtymologyNerd.))

        (SETQ posKeys (MergeKeywords (for word in posWords collect (APPLY* GetEntryTokensFn wordNerd word
                                                                          dictionary analyzer))
                              negKeys))
        (WordNerd.WeightedSearch wordNerd posKeys minWord maxWord T])
```

(MergeKeywords

```
  [LAMBDA (posWordLists negKeywords minimumMatches negWordLists) (* jtm%: " 1-Aug-88 15:11")
    (LET (intersection minimum n m order list)
         (OR minimumMatches (SETQ minimumMatches 2))
         [while posWordLists do (SETQ n 0)
                                (SETQ minimum NIL)
                                [for tail on posWordLists when (CAR tail)
                                   do (COND
                                         ((OR (NULL minimum)
                                              (ALPHORDER (CAAR tail)
                                                         minimum))
                                          (SETQ minimum (CAAR tail]
                                (OR minimum (RETURN))
                                [for tail on posWordLists when (CAR tail)
                                   do (while (EQUAL minimum (CAAR tail)) do (add n 1)
                                                                            (pop (CAR tail]
                                (COND
                                   ([AND (NOT (MEMBER minimum negKeywords))
                                         (OR (IGEQ n minimumMatches)
                                             (NULL (CDR posWordLists]
                                    (push intersection (LIST minimum n]
        intersection])

)

(DEFINEQ
```

(WORDNERD.PARSEINPUT

```
  [LAMBDA (INVERTEDDICT STRING IGNOREPARENS)                    (* jtm%: "12-Aug-88 16:45")
    (LET (ANALYZER KEYS SUBKEYS ENDPOS SUBSTRING (STARTPOS 1)
                (NCHARS (NCHARS STRING)))
         (SETQ ANALYZER (InvertedDict.Prop INVERTEDDICT 'ANALYZER))
         (COND
            ((NULL ANALYZER)
             (SETQ ANALYZER (create Morphalyzer))
             (InvertedDict.Prop INVERTEDDICT 'ANALYZER ANALYZER)))
         [while STARTPOS do (OR IGNOREPARENS (SETQ ENDPOS (STRPOS "(" STRING STARTPOS)))
                            [SETQ SUBSTRING (SUBSTRING STRING STARTPOS (SUB1 (OR ENDPOS (ADD1 NCHARS]
                            [AND SUBSTRING (Analyzer.Analyze ANALYZER SUBSTRING NIL NIL
                                                             (FUNCTION (LAMBDA (ANALYZER STREAM START LENGTH ENTRY)
                                                                               (push KEYS (OR ENTRY
                                                                                              (STREAM.FETCHSTRING STREAM START
                                                                                                     LENGTH NIL T)))
                                                                               NIL]
                            (COND
                               [ENDPOS (SETQ STARTPOS (ADD1 ENDPOS))
                                       (SETQ ENDPOS (STRPOS ")" STRING STARTPOS))
                                       (SETQ SUBKEYS NIL)
                                       [Analyzer.Analyze ANALYZER [SUBSTRING STRING STARTPOS
                                                                             (SUB1 (OR ENDPOS (ADD1 NCHARS]
                                              NIL NIL (FUNCTION (LAMBDA (ANALYZER STREAM START LENGTH ENTRY)
                                                                        (push SUBKEYS
                                                                              (OR ENTRY (STREAM.FETCHSTRING STREAM START
                                                                                                LENGTH NIL T)))
                                                                        NIL]
                                       (push KEYS (DREVERSE SUBKEYS))
                                       (SETQ STARTPOS (ADD1 (OR ENDPOS NCHARS]
                               (T (SETQ STARTPOS NIL]
         (SETQ KEYS (DREVERSE KEYS])

)

(DECLARE%: EVAL@COMPILE

(RECORD WNKEYINFO (WNKEY WNKEYWEIGHT WNKEYDATA))
)


         (* * SimpleNerd is an in-core version of the WordNerd.)

(DEFINEQ
```

(SimpleNerd.Create

```
  [LAMBDA (NAME DICTIONARY ANALYZER)                            ; Edited 25-Oct-88 12:01 by jtm:
    (LET (SIMPLENERD)
         (SETQ SIMPLENERD (create INVERTEDDICT
```

```
                              INVERTEDDICTNAME _ NAME))
        (AND DICTIONARY (InvertedDict.Prop SIMPLENERD 'DICTIONARY DICTIONARY))
        (InvertedDict.Prop SIMPLENERD 'ANALYZER (OR ANALYZER (SimpleAnalyzer.Create NAME)))
        (InvertedDict.Prop SIMPLENERD 'ADDASSOCIATIONFN (FUNCTION SimpleNerd.AddAssociation))
        (InvertedDict.Prop SIMPLENERD 'GETENTRYFN (FUNCTION SimpleNerd.GetEntry))
        (InvertedDict.Prop SIMPLENERD 'MAXHEADERIDFN (FUNCTION SimpleNerd.MaxEntry))
        (InvertedDict.Prop SIMPLENERD 'GETHEADERFN (FUNCTION SimpleNerd.GetHeader))
        (InvertedDict.Prop SIMPLENERD 'GETENTRYTOKENSFN (FUNCTION SimpleNerd.ParseDictEntry))
        (InvertedDict.Prop SIMPLENERD 'MAPKEYSFN (FUNCTION SimpleNerd.MapKeys))
        (InvertedDict.Prop SIMPLENERD 'VENNSEARCHFN (FUNCTION WordNerd.DefaultVennSearch))
        (InvertedDict.Prop SIMPLENERD 'WEIGHTEDSEARCHFN (FUNCTION WordNerd.DefaultWeightedSearch))
        (InvertedDict.Prop SIMPLENERD 'RELEVANCESEARCHFN (FUNCTION WordNerd.DefaultRelevanceSearch))
        (InvertedDict.Establish SIMPLENERD)
        SIMPLENERD])
```

(SimpleNerd.AddAssociation

```
[LAMBDA (WORDNERD HEADER KEY)                          ; Edited 21-Sep-88 14:37 by jtm:

        (* * adds KEY to WORDNERD under HEADER.)

  (LET (LASTENTRY HEADERINDEX INDEXFILE HEADERSIZE ENTRYID ASSOCS)

        (* * initialize local variables and data structures.)

       (COND
          ((NULL (SETQ HEADERINDEX (fetch (INVERTEDDICT HEADERINDEX) of WORDNERD)))
           (SETQ HEADERINDEX (ARRAY 100))
           (replace (INVERTEDDICT HEADERINDEX) of WORDNERD with HEADERINDEX)))
       (COND
          ((NULL (SETQ INDEXFILE (fetch (INVERTEDDICT INDEXFILE) of WORDNERD)))
           (SETQ INDEXFILE (SimpleDict.New (fetch (INVERTEDDICT INVERTEDDICTNAME) of WORDNERD)))
           (replace (INVERTEDDICT INDEXFILE) of WORDNERD with INDEXFILE)))
       [COND
          ([NULL (SETQ LASTENTRY (InvertedDict.Prop WORDNERD 'LASTENTRY]
           (SETQ LASTENTRY (CONS NIL 0]

        (* * map HEADER to a unique ID using EQUAL. We want ID numbers to make it easier to convert to external indices.)

       (COND
          ((EQ HEADER :STOPWORD)                       (* do nothing)
           NIL)
          ((EQUAL HEADER (CAR LASTENTRY))              (* this is optimized for multiple additions to the same entry.)
           (SETQ ENTRYID (CDR LASTENTRY)))
          ([SETQ ENTRYID (for I from 1 to (SETQ HEADERSIZE (ARRAYSIZE HEADERINDEX))
                            thereis (EQUAL HEADER (ELT HEADERINDEX I]
                                                       (* look for an existing entry.)
           (InvertedDict.Prop WORDNERD 'LASTENTRY (CONS HEADER ENTRYID)))
          (T                                           (* add an HEADER to HEADERINDEX)
             (SETQ ENTRYID (ADD1 (OR (InvertedDict.Prop WORDNERD 'LASTINDEX)
                                     0)))
             [COND
                ((IGREATERP ENTRYID HEADERSIZE)
                 (LET (NEWHEADERINDEX)                 (* get a bigger array.)
                      (SETQ NEWHEADERINDEX (ARRAY (ITIMES HEADERSIZE 2)))
                      (for I from 1 to HEADERSIZE do (SETA NEWHEADERINDEX I (ELT HEADERINDEX I)))
                      (replace (INVERTEDDICT HEADERINDEX) of WORDNERD with NEWHEADERINDEX)
                      (SETQ HEADERINDEX NEWHEADERINDEX]
             (SETA HEADERINDEX ENTRYID HEADER)
             (InvertedDict.Prop WORDNERD 'LASTINDEX ENTRYID)))

        (* * push the HEADER onto INDEXFILE)
                                                       (* used to #.(SEDIT::MAKE-BROKEN-ATOM "be:")
                                                       (SimpleDict.PushEntry INDEXFILE KEY ENTRYID
                                                       (QUOTE NEWTOP)))
       (COND
          ((EQ HEADER :STOPWORD)                       (* mark as stop word)
           (SimpleDict.PutEntry INDEXFILE KEY :STOPWORD))
          [(SETQ ASSOCS (SimpleDict.Lookup INDEXFILE KEY))
           (COND
              ((EQ ASSOCS :STOPWORD)                   (* don't do anything)
               NIL)
              ((NEQ ENTRYID (CAR ASSOCS))
               (ATTACH ENTRYID ASSOCS]
          (T (SimpleDict.PutEntry INDEXFILE KEY (LIST ENTRYID])
```

(SimpleNerd.MapKeys

```
[LAMBDA (NERD MAPFN)                                   ; Edited 25-Oct-88 11:57 by jtm:
  ;; Map through all of the keys in the NERD
  (Dict.MapEntries (fetch (INVERTEDDICT INDEXFILE) of NERD)
         (FUNCTION (LAMBDA (DICT PATH ENTRY)
                     (DECLARE (SPECVARS MAPFN NERD))
                     (LET (HEADER)
                          (SETQ HEADER (CONCATLIST PATH))      ; PATH is a list of characters
                          (APPLY* MAPFN NERD HEADER HEADER ENTRY))
```

```
                                                                    ; KEY and KEYID are the same.  ENTRY may be :STOPWORD
                        ])
```

(SimpleNerd.GetEntry

```
  [LAMBDA (WORDNERD KEYID INDEXFILE)                                (* jtm%: " 2-Aug-88 10:13")
    (SimpleDict.Lookup INDEXFILE KEYID])
```

(SimpleNerd.MaxEntry

```
  [LAMBDA (WORDNERD HEADERINDEX)                                    (* jtm%: " 1-Aug-88 16:44")
    (InvertedDict.Prop WORDNERD 'LASTINDEX])
```

(SimpleNerd.GetHeader

```
  [LAMBDA (WORDNERD HEADERID HEADERINDEX BUFFER)                    ; Edited 11-Nov-88 14:30 by jtm:
    (COND
       (HEADERINDEX (ELT HEADERINDEX HEADERID))
       (T HEADERID])
```

(SimpleNerd.ParseDictEntry

```
  [LAMBDA (WORDNERD WORD DICTIONARY ANALYZER)                       ; Edited 24-Oct-88 14:19 by jtm:

          (* * return the list of tokens in the definition of WORD.)

    (LET (DICTENTRY TOKENS HARRAY)
         [OR DICTIONARY (SETQ DICTIONARY (InvertedDict.Prop WORDNERD 'DICTIONARY]
         [OR ANALYZER (SETQ ANALYZER (InvertedDict.Prop WORDNERD 'ANALYZER]
         (SETQ DICTENTRY (Dict.GetEntry DICTIONARY WORD))
         [COND
            (DICTENTRY (SETQ HARRAY (HASHARRAY 100 NIL 'STRINGHASHBITS 'STREQUAL))
                   (Analyzer.Analyze ANALYZER DICTENTRY NIL NIL
                          (FUNCTION (LAMBDA (ANALYZER STREAM START LENGTH ENTRY)
                                    (LET (TOKEN)
                                         (SETQ TOKEN (OR ENTRY (STREAM.FETCHSTRING STREAM START LENGTH NIL T))
                                          )
                                         (PUTHASH TOKEN T HARRAY)
                                                                    ; return NIL to keep iteration going
                                        NIL]
         (COND
            ((STREAMP DICTENTRY)
             (CLOSEF DICTENTRY)))
         [MAPHASH HARRAY (FUNCTION (LAMBDA (VAL KEY)
                                   (push TOKENS KEY]
         TOKENS])
```

(SimpleNerd.Test

```
  [LAMBDA NIL                                                       (* jtm%: " 2-Aug-88 14:54")
    (LET (simpleNerd simpleDict analyzer GetEntryTokensFn)
         (SETQ simpleDict (SimpleDict.New 'TEST))
         (Dict.PutEntry simpleDict "Paine" "Now is the time for all good men to come to the aid of their
                country.")
         (Dict.PutEntry simpleDict "Jefferson" "Now is the time for good men to help out.")
         (Dict.PutEntry simpleDict "King George" "Now wait a minute!")
         (Dict.PutEntry simpleDict "Kennedy" "Ask not what your country can do for you.")
         (replace (Dict printEntryFn) of simpleDict with (FUNCTION DictTool.PrintDefinition))
         (Dict.Establish simpleDict)
         (SETQ simpleNerd (SimpleNerd.Create 'TEST))
         (InvertedDict.Prop simpleNerd 'DICTIONARY simpleDict)
         (InvertedDict.Prop simpleNerd 'MINIMUMWEIGHT 0)
         (SETQ analyzer (InvertedDict.Prop simpleNerd 'ANALYZER))
         (SETQ GetEntryTokensFn (InvertedDict.Prop simpleNerd 'GETENTRYTOKENSFN))
         [Dict.MapEntries simpleDict (FUNCTION (LAMBDA (dict path value)
                                               (LET (string tokens)
                                                    (SETQ string (CONCATLIST path))
                                                    (SETQ tokens (APPLY* GetEntryTokensFn simpleNerd string
                                                                         simpleDict analyzer))
                                                    (for token in tokens do (SimpleNerd.AddEntry simpleNerd
                                                                                   string token]
         (SETQ TESTNERD simpleNerd])

)

          (* * HashfileNerd stores its data structures in an InterLisp hashfile.)

(DEFINEQ

(HashfileNerd.Create
  [LAMBDA (NAME FILENAME DICTIONARY ANALYZER)                       ; Edited 17-Nov-88 17:03 by jtm:
    (LET (NERD)
         (SETQ NERD (create INVERTEDDICT
                           INVERTEDDICTNAME _ NAME
                           HEADERINDEX _ (ARRAY 100)))
```

```
;; HashfileNerd.AddAssociation used to create the header array on demand, but now I want a NIL header index to indicate the identity
;; mapping, so the array is created here and removed if the user wants an identity mapping.
(InvertedDict.Prop NERD 'FILENAME FILENAME)
(InvertedDict.Prop NERD 'ANALYZER (OR ANALYZER (SimpleAnalyzer.Create NAME)))
(COND
   (DICTIONARY (InvertedDict.Prop NERD 'DICTIONARY DICTIONARY)))
(InvertedDict.Prop NERD 'OPENFN (FUNCTION HashfileNerd.Open))
(InvertedDict.Prop NERD 'CLOSEFN (FUNCTION HashfileNerd.Close))
(InvertedDict.Prop NERD 'RELEVANCESEARCHFN (FUNCTION WordNerd.DefaultRelevanceSearch))
(InvertedDict.Prop NERD 'WEIGHTEDSEARCHFN (FUNCTION WordNerd.DefaultWeightedSearch))
(InvertedDict.Prop NERD 'VENNSEARCHFN (FUNCTION WordNerd.DefaultVennSearch))
(InvertedDict.Prop NERD 'ADDASSOCIATIONFN (FUNCTION HashfileNerd.AddAssociation))
(InvertedDict.Prop NERD 'GETENTRYFN (FUNCTION HashfileNerd.GetEntry))
(InvertedDict.Prop NERD 'MAXHEADERIDFN (FUNCTION SimpleNerd.MaxEntry))
(InvertedDict.Prop NERD 'GETHEADERFN (FUNCTION SimpleNerd.GetHeader))
(InvertedDict.Prop NERD 'MAPKEYSFN (FUNCTION HashfileNerd.MapKeys))
(InvertedDict.Prop NERD 'EXPANDKEYPATTERNFN (FUNCTION HashfileNerd.ExpandKeyPattern))
(InvertedDict.Prop NERD 'GETENTRYTOKENSFN (FUNCTION SimpleNerd.ParseDictEntry))
(InvertedDict.Establish NERD)
NERD])
```

(HashfileNerd.Test

```
[LAMBDA (FILEPATTERN FILENAME)                                  ; Edited 26-Sep-88 13:30 by jtm:
  (LET (NERD DICT)
       (SETQ DICT (FileDict.Create 'TEST))
       (FileDict.AddFiles DICT FILEPATTERN)
       (SETQ NERD (HashfileNerd.Create 'TEST NIL DICT))
       (WordNerd.AddDictionary NERD DICT)
       (HashfileNerd.Write NERD FILENAME)
       (HashfileNerd.Close NERD)
       (SETQ NERD (HashfileNerd.Create 'TEST NIL DICT))
       (HashfileNerd.Open NERD FILENAME)
       NERD])
```

(HashfileNerd.Open

```
[LAMBDA (WORDNERD FILENAME)                                     ; Edited 11-Nov-88 11:34 by jtm:
```

;;; Reads a hashfilenerd out of the hashfile stored in FILENAME

```
  (LET (HASHFILE HEADERLIST HEADERINDEX DICTNAME ANALYZERNAME)
       (COND
          ([AND (NULL (fetch (INVERTEDDICT INDEXFILE) of WORDNERD))
                (OR FILENAME (SETQ FILENAME (InvertedDict.Prop WORDNERD 'FILENAME]
           (SETQ HASHFILE (OPENHASHFILE FILENAME))
           ;; read out the name
           (replace (INVERTEDDICT INVERTEDDICTNAME) of WORDNERD with (OR (GETHASHFILE '*NAME* HASHFILE)
                                                                         (fetch (INVERTEDDICT INVERTEDDICTNAME)
                                                                            of WORDNERD)))
           ;; read out the HEADERINDEX
           [COND
              ((SETQ HEADERLIST (GETHASHFILE '*HEADERINDEX* HASHFILE))
               [SETQ HEADERINDEX (ARRAY (IPLUS 10 (LENGTH HEADERLIST]
               (for I in HEADERLIST do (SETA HEADERINDEX (CAR I)
                                             (CADR I]
           ;; read out simple properties
           (for PROP in (GETHASHFILE '*PROPS* HASHFILE) do (InvertedDict.Prop WORDNERD (CAR PROP)
                                                                              (CDR PROP)))
           (replace (INVERTEDDICT HEADERINDEX) of WORDNERD with HEADERINDEX)
           ;; check the dictionary name
           (COND
              ([AND (SETQ DICTNAME (GETHASHFILE '*DICTIONARY* HASHFILE))
                    (NEQ DICTNAME (Dict.Name (InvertedDict.Prop WORDNERD 'DICTIONARY]
               (ERROR "WORDNERD has wrong dictionary for this hash file.")))
           ;;  check the analyzer name
           (COND
              ([AND (SETQ ANALYZERNAME (GETHASHFILE '*ANALYZER* HASHFILE))
                    (NEQ ANALYZERNAME (Analyzer.Name (InvertedDict.Prop WORDNERD 'ANALYZER]
               (ERROR "WORDNERD has wrong analyzer for this hash file.")))
           ;; finally replace INDEXFILE with HASHFILE atomically
           (replace (INVERTEDDICT INDEXFILE) of WORDNERD with HASHFILE])
```

(HashfileNerd.Close

```
[LAMBDA (WORDNERD)                                              ; Edited 15-Sep-88 10:04 by jtm:
  (LET (INDEXFILE)
       (SETQ INDEXFILE (fetch (INVERTEDDICT INDEXFILE) of WORDNERD))
       [COND
          ((OR (HARRAYP INDEXFILE)
```

```
            (LISTP INDEXFILE))
        (HashfileNerd.Write WORDNERD)
        (SETQ INDEXFILE (fetch (INVERTEDDICT INDEXFILE) of WORDNERD]
    (COND
       ((HASHFILEP INDEXFILE)
        (CLOSEHASHFILE INDEXFILE)
        (replace (INVERTEDDICT INDEXFILE) of WORDNERD with NIL)
        (InvertedDict.Prop WORDNERD 'LASTINDEX NIL)
        (InvertedDict.Prop WORDNERD 'LASTENTRY NIL)))
    NIL])
```

(HashfileNerd.Write

```
  [LAMBDA (WORDNERD FILENAME)                                   ; Edited 11-Nov-88 11:32 by jtm:
```

;;; Write out WORDNERD onto a hashfile so that it can be read back in later.

```
(LET (INDEXFILE HEADERINDEX HASHFILE HASHARRAY PROPS DICT DICTNAME ANALYZER ANALYZERNAME NAME)
     ;; defaults for FILENAME
     (COND
        [(NULL FILENAME)
         (SETQ FILENAME (InvertedDict.Prop WORDNERD 'FILENAME]
        (T (InvertedDict.Prop WORDNERD 'FILENAME FILENAME)))
     (OR FILENAME (ERROR "Please specify a filename."))
     ;; set up HASHFILE and HASHARRAY
     (SETQ INDEXFILE (fetch (INVERTEDDICT INDEXFILE) of WORDNERD))
     (COND
        ((LISTP INDEXFILE)                                      ; information was added to an existing hashfile
         (SETQ HASHARRAY (CAR INDEXFILE))
         (SETQ HASHFILE (CDR INDEXFILE)))
        ((HASHFILEP INDEXFILE)
         (SETQ HASHFILE INDEXFILE))
        ((HARRAYP INDEXFILE)
         (SETQ HASHARRAY INDEXFILE))
        ((BIGHASHP INDEXFILE)
         (SETQ HASHARRAY INDEXFILE)))
     [COND
        ((NULL HASHFILE)                                        ; create a new hashfile
         (SETQ HASHFILE (CREATEHASHFILE FILENAME NIL NIL (BIGHASHSIZE HASHARRAY]
     [COND
        (HASHARRAY                                              ; write new entries out
               (BIGMAPHASH HASHARRAY (FUNCTION (LAMBDA (VAL KEY)
                                                 (PUTHASHFILE KEY VAL HASHFILE]
     ;; store the name on the hashfile if it has changed.
     (SETQ NAME (fetch (INVERTEDDICT INVERTEDDICTNAME) of WORDNERD))
     (COND
        ((NEQ NAME (GETHASHFILE '*NAME* HASHFILE))
         (PUTHASHFILE '*NAME* NAME HASHFILE)))
     ;; store HEADERINDEX on the hashfile if it has changed.
     (SETQ HEADERINDEX (fetch (INVERTEDDICT HEADERINDEX) of WORDNERD))
     [SETQ HEADERLIST (AND HEADERINDEX (for I ENTRY from 1 while (SETQ ENTRY (ELT HEADERINDEX I))
                                          collect (LIST I ENTRY]
     (COND
        ((NOT (EQUAL HEADERLIST (GETHASHFILE '*HEADERINDEX* HASHFILE)))
         (PUTHASHFILE '*HEADERINDEX* HEADERLIST HASHFILE)))
     ;; store simple properties on the hashfile if they have changed.
     (SETQ PROPS (for PROP in (fetch (INVERTEDDICT INVERTEDDICTPROPS) of WORDNERD)
                    when (SIMPLETYPE (CDR PROP)) collect PROP))
     (COND
        ((NOT (EQUAL PROPS (GETHASHFILE '*PROPS* HASHFILE)))
         (PUTHASHFILE '*PROPS* PROPS HASHFILE)))
     ;; store the name of the dictionary on the hashfile if it hash changed.
     (SETQ DICTNAME (AND [type? Dict (SETQ DICT (InvertedDict.Prop WORDNERD 'DICTIONARY]
                         (Dict.Name DICT)))
     (COND
        ((NEQ DICTNAME (GETHASHFILE '*DICTIONARY* HASHFILE))
         (PUTHASHFILE '*DICTIONARY* DICTNAME HASHFILE)))
     ;; store the name of the analyzer if it has changed.
     (SETQ ANALYZERNAME (AND [type? Morphalyzer (SETQ ANALYZER (InvertedDict.Prop WORDNERD 'DICTIONARY]
                             (Analyzer.Name ANALYZER)))
     (COND
        ((NEQ ANALYZERNAME (GETHASHFILE '*ANALYZER* HASHFILE))
         (PUTHASHFILE '*ANALYZER* ANALYZERNAME HASHFILE)))
     ;; remove the hash array from INDEXFILE
     (replace (INVERTEDDICT INDEXFILE) of WORDNERD with HASHFILE)
     ;; return the hash file
     HASHFILE])
```

(SIMPLETYPE

```
  [LAMBDA (DATUM)                                          ; Edited 15-Sep-88 14:33 by jtm:
    (COND
       ((NUMBERP DATUM)
        T)
       ((STRINGP DATUM)
        T)
       ((LITATOM DATUM)
        T)
       ((LISTP DATUM)
        (for I inside DATUM always (SIMPLETYPE I])
```

(HashfileNerd.AddAssociation

```
  [LAMBDA (WORDNERD HEADER KEY)                            ; Edited 11-Nov-88 11:30 by jtm:

          (* * adds KEY to WORDNERD under HEADER.)

    (LET (LASTENTRY HEADERINDEX INDEXFILE HEADERSIZE ENTRYID HASHARRAY HASHFILE ASSOCS)

          (* * initialize local variables and data structures.)

        (WordNerd.Open WORDNERD)
        (SETQ HEADERINDEX (fetch (INVERTEDDICT HEADERINDEX) of WORDNERD))
        (SETQ INDEXFILE (fetch (INVERTEDDICT INDEXFILE) of WORDNERD))
        [COND
           ((HARRAYP INDEXFILE)
            (SETQ HASHARRAY INDEXFILE))
           ((BIGHASHP INDEXFILE)
            ;; INDEXFILE is an array of hash arrays indexed by length of the key string.
            (SETQ HASHARRAY INDEXFILE))
           ((HASHFILEP INDEXFILE)
            (SETQ HASHFILE INDEXFILE)
            ;; Create a hash array to handle the new associations.
            (SETQ HASHARRAY (HASHARRAY 100 NIL (FUNCTION STRINGHASHBITS)
                                   (FUNCTION STREQUAL)))
            (SETQ INDEXFILE (CONS HASHARRAY HASHFILE))
            (replace (INVERTEDDICT INDEXFILE) of WORDNERD with INDEXFILE))
           ((NULL INDEXFILE)
            (SETQ HASHARRAY (HASHARRAY 100 NIL (FUNCTION STRINGHASHBITS)
                                   (FUNCTION STREQUAL)))
            (replace (INVERTEDDICT INDEXFILE) of WORDNERD with HASHARRAY))
           ((LISTP INDEXFILE)                                ; must follow (NULL INDEXFILE)
            (SETQ HASHARRAY (CAR INDEXFILE))
            (SETQ HASHFILE (CDR INDEXFILE]
        [COND
           ([NULL (SETQ LASTENTRY (InvertedDict.Prop WORDNERD 'LASTENTRY]
            (SETQ LASTENTRY (CONS NIL 0]

          (* * map HEADER to a unique ID using EQUAL. We want ID numbers to make it easier to convert to external indices.)

        (COND
           ((EQ HEADER :STOPWORD)                           (* don't add header to header index.)
            NIL)
           ((NULL HEADERINDEX)                              (* Identity mapping)
            (SETQ ENTRYID HEADER))
           ((EQUAL HEADER (CAR LASTENTRY))                  (* this is optimized for multiple additions to the same entry.)
            (SETQ ENTRYID (CDR LASTENTRY)))
           ([SETQ ENTRYID (for I from 1 to (SETQ HEADERSIZE (ARRAYSIZE HEADERINDEX))
                             thereis (EQUAL HEADER (ELT HEADERINDEX I]
                                                            (* look for an existing entry.)
            (InvertedDict.Prop WORDNERD 'LASTENTRY (CONS HEADER ENTRYID)))
           (T                                               (* add an HEADER to HEADERINDEX)
              (SETQ ENTRYID (ADD1 (OR (InvertedDict.Prop WORDNERD 'LASTINDEX)
                                      0)))
              [COND
                 ((IGREATERP ENTRYID HEADERSIZE)
                  (LET (NEWHEADERINDEX)                     (* get a bigger array.)
                       (SETQ NEWHEADERINDEX (ARRAY (ITIMES HEADERSIZE 2)))
                       (for I from 1 to HEADERSIZE do (SETA NEWHEADERINDEX I (ELT HEADERINDEX I)))
                       (replace (INVERTEDDICT HEADERINDEX) of WORDNERD with NEWHEADERINDEX)
                       (SETQ HEADERINDEX NEWHEADERINDEX]
              (SETA HEADERINDEX ENTRYID HEADER)
              (InvertedDict.Prop WORDNERD 'LASTINDEX ENTRYID)))

          (* * push the HEADER onto INDEXFILE)

        (COND
           ((EQ HEADER :STOPWORD)
            (BIGPUTHASH KEY :STOPWORD HASHARRAY))
           [(SETQ ASSOCS (BIGGETHASH KEY HASHARRAY))
            (COND
               ((EQ ASSOCS :STOPWORD)                       (* don't do anything)
                NIL)
```

```
               ((NEQ ENTRYID (CAR ASSOCS))
                (ATTACH ENTRYID ASSOCS]
            ((AND HASHFILE (SETQ ASSOCS (GETHASHFILE KEY HASHFILE))
                  (NEQ ASSOCS :STOPWORD)
                  (NEQ ENTRYID (CAR ASSOCS)))
             (push ASSOCS ENTRYID)
             (BIGPUTHASH KEY ASSOCS HASHARRAY))
            (T (BIGPUTHASH KEY (LIST ENTRYID)
                      HASHARRAY])
```

(HashfileNerd.GetEntry

```
  [LAMBDA (WORDNERD KEYID INDEXFILE)                          ; Edited 11-Nov-88 14:28 by jtm:
    [COND
       ((NULL INDEXFILE)
        (SETQ INDEXFILE (fetch (INVERTEDDICT INDEXFILE) of WORDNERD]
    (COND
       ((BIGHASHP INDEXFILE)
        (BIGGETHASH KEYID INDEXFILE))
       [(LISTP INDEXFILE)                                     (* (CONS HASHARRAY HASHFILE))
        (COND
           ((GETHASH KEYID (CAR INDEXFILE)))
           (T (GETHASHFILE KEYID (CDR INDEXFILE]
       ((HASHFILEP INDEXFILE)
        (GETHASHFILE KEYID INDEXFILE))
       ((HARRAYP INDEXFILE)
        (GETHASH KEYID INDEXFILE])
```

(HashfileNerd.ExpandKeyPattern

```
  [LAMBDA (NERD KEYPATTERN)                                   ; Edited 17-Nov-88 16:58 by jtm:
    (LET (PAT INDEXFILE KEYS HASHARRAY HASHFILE)
         (SETQ PAT (DIRECTORY.MATCH.SETUP KEYPATTERN))
         (SETQ INDEXFILE (fetch (INVERTEDDICT INDEXFILE) of NERD))
         [COND
            ((OR (BIGHASHP INDEXFILE)
                 (HARRAYP INDEXFILE))
             (SETQ HASHARRAY INDEXFILE))
            ((HASHFILEP INDEXFILE)
             (SETQ HASHFILE INDEXFILE))
            ((LISTP INDEXFILE)
             (SETQ HASHARRAY (CAR INDEXFILE))
             (SETQ HASHFILE (CDR INDEXFILE]
         [if HASHARRAY
             then (BIGMAPHASH HASHARRAY (FUNCTION (LAMBDA (DATA KEY)
                                                    (COND
                                                       ((DIRECTORY.MATCH PAT KEY)
                                                        (push KEYS KEY]
         [if HASHFILE
             then (MAPHASHFILE HASHFILE (FUNCTION (LAMBDA (DATA KEY)
                                                   (COND
                                                      ((DIRECTORY.MATCH PAT KEY)
                                                       (push KEYS KEY]
         KEYS])
```

(HashfileNerd.MapKeys

```
  [LAMBDA (NERD KEYFN)                                        ; Edited 17-Nov-88 16:29 by jtm:
```

;;; maps through all of the keys in the hash array/ file

```
    (LET (INDEXFILE HASHARRAY HASHFILE)
         (SETQ INDEXFILE (fetch (INVERTEDDICT INDEXFILE) of NERD))
         [COND
            ((OR (BIGHASHP INDEXFILE)
                 (HARRAYP INDEXFILE))
             (SETQ HASHARRAY INDEXFILE))
            ((HASHFILEP INDEXFILE)
             (SETQ HASHFILE INDEXFILE))
            ((LISTP INDEXFILE)
             (SETQ HASHARRAY (CAR INDEXFILE))
             (SETQ HASHFILE (CDR INDEXFILE]
         [if HASHARRAY
             then (BIGMAPHASH HASHARRAY (FUNCTION (LAMBDA (DATA KEY)
                                                    (APPLY* KEYFN NERD KEY DATA]
         (if HASHFILE
             then (MAPHASHFILE HASHFILE (FUNCTION (LAMBDA (DATA KEY)
                                                   (APPLY* KEYFN NERD KEY DATA])

)

(DEFINEQ
```

(BIGHASH

```
  [LAMBDA (MAXKEYLENGTH MINKEYS OVERFLOW HASHBITSFN EQUIVFN)  ; Edited 11-Nov-88 11:19 by jtm:
```

;; A BIGHASH first hashes keys into an array by key length to get a hash array and then hashes the key in that hash array.

```
    (create BIGHASH
           BIGHASHARRAY _ (ARRAY (ADD1 MAXKEYLENGTH)
                                 NIL NIL 0)
           BIGHASHMINKEYS _ MINKEYS
           BIGHASHOVERFLOW _ OVERFLOW
           BIGHASHBITSFN _ HASHBITSFN
           BIGHASHEQUIVFN _ EQUIVFN])
```

(BIGGETHASH

```
  [LAMBDA (KEY BIGHASH)                                    ; Edited 11-Nov-88 11:04 by jtm:
    ;; get the value from the hash array that has keys of KEY's length
    (LET (HARRAY)
         (COND
            [(type? BIGHASH BIGHASH)
             (COND
                ((SETQ HARRAY (ELT (fetch BIGHASHARRAY of BIGHASH)
                                   (NCHARS KEY)))
                 (GETHASH KEY HARRAY]
            (T (GETHASH KEY BIGHASH])
```

(BIGHASHSIZE

```
  [LAMBDA (BIGHASH)                                        ; Edited 11-Nov-88 11:28 by jtm:
    (LET (ARRAY)
         (COND
            [(type? BIGHASH BIGHASH)
             (SETQ ARRAY (fetch BIGHASHARRAY of BIGHASH))
             (for I HARRAY from 0 to (SUB1 (ARRAYSIZE ARRAY)) sum (COND
                                                                     ((SETQ HARRAY (ELT ARRAY I))
                                                                      (HARRAYSIZE HARRAY))
                                                                     (T 0]
            (T (HARRAYSIZE BIGHASH])
```

(BIGMAPHASH

```
  [LAMBDA (BIGHASH MAPBHFN)                                ; Edited 11-Nov-88 11:10 by jtm:
    ;; map through all of the keys in the sub-hashes.
    (LET (ARRAY)
         (COND
            [(type? BIGHASH BIGHASH)
             (SETQ ARRAY (fetch BIGHASHARRAY of BIGHASH))
             (for I HARRAY from 0 to (SUB1 (ARRAYSIZE ARRAY)) do (COND
                                                                    ((SETQ HARRAY (ELT ARRAY I))
                                                                     (MAPHASH HARRAY MAPBHFN]
            (T (MAPHASH BIGHASH MAPBHFN])
```

(BIGPUTHASH

```
  [LAMBDA (KEY VAL BIGHASH)                                ; Edited 11-Nov-88 11:02 by jtm:
    ;; put all of the keys with the same lengths together.
    (LET (HARRAY ARRAY NCHARS)
         (COND
            ((type? BIGHASH BIGHASH)
             (SETQ NCHARS (NCHARS KEY))
             (SETQ ARRAY (fetch BIGHASHARRAY of BIGHASH))
             (SETQ HARRAY (ELT ARRAY NCHARS))
             (COND
                ((NULL HARRAY)
                 (SETQ HARRAY (HASHARRAY (fetch BIGHASHMINKEYS of BIGHASH)
                                         (fetch BIGHASHOVERFLOW of BIGHASH)
                                         (fetch BIGHASHBITSFN of BIGHASH)
                                         (fetch BIGHASHEQUIVFN of BIGHASH)))
                 (SETA ARRAY NCHARS HARRAY)))
             (PUTHASH KEY VAL HARRAY))
            (T (PUTHASH KEY VAL BIGHASH])

)

(DECLARE%: EVAL@COMPILE

(PUTPROPS BIGHASHP MACRO ((ARRAY)
                          (type? BIGHASH ARRAY)))
)

(DECLARE%: EVAL@COMPILE

(TYPERECORD BIGHASH (BIGHASHARRAY BIGHASHMINKEYS BIGHASHOVERFLOW BIGHASHBITSFN BIGHASHEQUIVFN))
)


          (* * the following should be merged into ANALYZER eventually.)

(DEFINEQ
```

(**FileDict.Create**
```
  [LAMBDA (NAME FILENAME)                                   ; Edited 15-Sep-88 14:02 by jtm:
    (LET (DICT)
         (SETQ DICT (SimpleDict.Create NAME FILENAME))
         (replace (Dict getEntryFn) of DICT with (FUNCTION FileDict.Lookup))
         (replace (Dict mapFn) of DICT with (FUNCTION FileDict.MapEntries))
         (replace (Dict printEntryFn) of DICT with (FUNCTION FileDict.PrintEntry))
         (Dict.Establish DICT)
        DICT])
```

(**FileDict.AddFiles**
```
  [LAMBDA (DICT FILEPATTERN)                                ; Edited 14-Sep-88 16:30 by jtm:
    (for FULLNAME in (SORT (DIRECTORY FILEPATTERN)) do (Dict.PutEntry DICT (FILENAMEFIELD FULLNAME 'NAME)
                                                              FULLNAME])
```

(**FileDict.PrintEntry**
```
  [LAMBDA (DICT KEY STREAM)                                 ; Edited 14-Sep-88 16:38 by jtm:
    (LET (FILE)
         (COND
            ((SETQ FILE (SimpleDict.Lookup DICT KEY))
             (TEDIT FILE)
             ""])
```

(**FileDict.Write**
```
  [LAMBDA (DICT FILENAME)                                   ; Edited 15-Sep-88 11:13 by jtm:
    (LET (STREAM)
         (SETQ STREAM (OPENSTREAM FILENAME 'OUTPUT])
```

(**FileDict.Lookup**
```
  [LAMBDA (DICT KEY)                                        ; Edited 14-Sep-88 15:08 by jtm:
    (LET (FILE)
         (COND
            ((SETQ FILE (SimpleDict.Lookup DICT KEY))
             (FETCHSTRINGFROMFILE FILE])
```

(**FileDict.MapEntries**
```
  [LAMBDA (DICT FDMAPFN PROP)                               ; Edited 14-Sep-88 15:26 by jtm:
    (SimpleDict.MapEntries DICT (FUNCTION (LAMBDA (DICT KEY FILENAME)
                                          (LET (STREAM)
                                               (SETQ STREAM (OPENTEXTSTREAM FILENAME))
                                               (APPLY* FDMAPFN DICT (CONCATLIST KEY)
                                                       STREAM)
                                               (CLOSEF STREAM])
```

(**FETCHSTRINGFROMFILE**
```
  [LAMBDA (FILENAME)                                        ; Edited 15-Sep-88 11:01 by jtm:
                                                            (* LET (STREAM STRING) (SETQ STREAM
                                                            (OPENTEXTSTREAM FILENAME))
                                                            (TEDIT.SETSEL STREAM 1 10000
                                                            (QUOTE LEFT)) (SETQ STRING
                                                            (TEDIT.SEL.AS.STRING STREAM))
                                                            (CLOSEF STREAM))
    (COND
       ((LISTP FILENAME)
        (OPENTEXTSTREAM (CAR FILENAME)
               NIL
               (CADR FILENAME)
               (CADDR FILENAME)))
       (T (OPENTEXTSTREAM FILENAME])

)

(DEFINEQ
```

(**SimpleAnalyzer.Create**
```
  [LAMBDA (NAME)                                            ; Edited 14-Sep-88 09:49 by jtm:
    (LET (morphalyzer)
         (SETQ morphalyzer (create Morphalyzer
                                   analyzerName _ NAME
                                   lookupFn _ (FUNCTION SimpleAnalyzer.Lookup)))
         (Analyzer.Prop morphalyzer 'OPT-SEPR-CODES '(NIL))

          (* * turn off the optional seperator codes)

         morphalyzer])
```

(**SimpleAnalyzer.Lookup**
```
  [LAMBDA (ANALYZER STREAM START LENGTH)                    ; Edited 14-Sep-88 09:46 by jtm:
    (L-CASE (COND
```

```
               ((STRINGP STREAM)
                (SUBSTRING STREAM (ADD1 START)
                       (IPLUS START LENGTH)))
               (T (STREAM.FETCHSTRING STREAM START LENGTH NIL T])

)

(DEFINEQ
```

(SimpleDict.Create

```
  [LAMBDA (name filename)                                    ; Edited 15-Sep-88 11:26 by jtm:
    (LET (dict)
         (SETQ dict (create Dict
                           dictName _ name
                           openFn _ (FUNCTION SimpleDict.Open)
                           closeFn _ (FUNCTION SimpleDict.Close)
                           getEntryFn _ (FUNCTION SimpleDict.Lookup)
                           putEntryFn _ (FUNCTION SimpleDict.PutEntry)
                           mapFn _ (FUNCTION SimpleDict.MapEntries)
                           contents _ (create SimpleDict.Node)))
         (Dict.Prop dict 'FILENAME filename)
         dict])
```

(SimpleDict.Open

```
  [LAMBDA (DICT)                                             ; Edited 15-Sep-88 11:38 by jtm:
    (LET (FILENAME)
         (COND
            ([AND (NULL (fetch (SimpleDict.Node subnodes) of (fetch (Dict contents) of DICT)))
                  (SETQ FILENAME (Dict.Prop DICT 'FILENAME]
             (SETQ DATALIST (CDR (READFILE FILENAME)))
             (for PAIR in DATALIST do (Dict.PutEntry DICT (CAR PAIR)
                                                     (CADR PAIR])
```

(SimpleDict.Close

```
  [LAMBDA (DICT)                                             ; Edited 15-Sep-88 11:30 by jtm:
    (LET (CONTENTS FILENAME)
         (COND
            ([AND (SETQ CONTENTS (fetch (Dict contents) of DICT))
                  (fetch (SimpleDict.Node subnodes) of CONTENTS)
                  (SETQ FILENAME (Dict.Prop DICT 'FILENAME]
             (SimpleDict.Write DICT FILENAME)
             (replace (Dict contents) of DICT with (create SimpleDict.Node))
             DICT])
```

(SimpleDict.Write

```
  [LAMBDA (DICT FILENAME)                                    ; Edited 15-Sep-88 11:21 by jtm:
    (LET (DATALIST)
         (COND
            [(NULL FILENAME)
             (SETQ FILENAME (Dict.Prop DICT 'FILENAME]
            (T (Dict.Prop DICT 'FILENAME FILENAME)))
         [SimpleDict.MapEntries DICT (FUNCTION (LAMBDA (DICT PATH ENTRY)
                                                 (push DATALIST (LIST (CONCATLIST PATH)
                                                                      ENTRY]
         (WRITEFILE (DREVERSE DATALIST)
                FILENAME)
         DICT])

)

(RPAQQ ENGLISHSTOPWORDS
       ("from" "to" "the" "of" "in" "a" "for" "on" "and" "that" "s" "is" "at" "it" "be" "by" "with" "but" "an"
             "not" "i" "was" "as" "t" "this" "they" "you" "he" "all" "if" "who" "may" "have" "out" "or" "when"
             "are" "so" "his" "can" "which" "about" "had" "been" "were" "do" "has" "other" "would" "we" "also"
             "some" "your" "my" "me" "their" "no" "could" "only" "more" "then" "him" "our" "any" "them" "her"
             "over" "its" "before" "between" "what" "after" "she" "most" "those" "than" "these" "does" "same"
             "into" "such" "while" "here" "how" "off" "will" "around" "there"))

(PUTPROPS WORDNERD COPYRIGHT ("Xerox Corporation" 1988))
```

FUNCTION INDEX

MACRO INDEX

RECORD INDEX

VARIABLE INDEX