File created: 17-May-90 11:20:15 **{DSK}<usr>local>lde>lispcore>sources>SPELL.;2**

changes to: (VARS SPELLCOMS)

previous date: 15-Nov-86 22:33:41 {DSK}<usr>local>lde>lispcore>sources>SPELL.;1

Read Table: XCL

Package: INTERLISP

Format: XCCS


```
(RPAQQ SPELLCOMS
       ((FNS ADDSPELL ADDSPELL1 ADDSPELL2 MISSPELLED? FIXSPELL FIXSPELL1 FIXSPELL2 CHOOZ CHOOZ1 SETSPELLCASE
             SKOR0 SKOR MOVETOP)
        (INITVARS (USERWORDS)
               (SPELLINGS1 '(DEFINEQ ARGLIST MOVD GETD FNTYP BREAK UNBREAK REBREAK TRACE BREAKIN MAKEFILE
                                     MAKEFILES LISTFILES FILES? WHEREIS CLEANUP PP PF EDITF EDITV EDITP ADVISE
                                     UNADVISE UNSAVEDEF RECOMPILE TCOMPL COMPILE BRECOMPILE BCOMPL MAPCAR MAPC
                                     LOAD LOADFROM LOADFNS TIME CLOSEF CLOSEALL OPENP OUTPUT INPUT OUTFILE INFILE
                                     LOGOUT PUTPROP REMPROP GETPROP SYSOUT CLISPIFY DWIMIFY EDITCALLERS FREEVARS
                                     CALLS))
               (SPELLINGS2 '(GETPROP ADD1 AND APPEND ASSOC COND CONS COPY ELT EQ EQUAL ERROR ERSETQ EVAL FASSOC
                                     FMEMB FRPLACA FRPLACD FUNCTION GO IDIFFERENCE IGREATERP ILESSP IMINUS IPLUS
                                     ITIMES LENGTH LIST LISTP MAPC MAPCAR MAPCONC MEMB MEMBER NCONC NCONC1 NEQ
                                     NLISTP NLSETQ NULL NUMBERP OR PRINT PRIN1 PROG PROGN PUTPROP QUOTE READ
                                     RETURN RPLACA RPLACD SELECTQ SETA SETQ SPACES SUB1 TERPRI ZEROP IF F/L
                                     VALUEOF FOR FETCH REPLACE CREATE GETPROP PUTPROP DIFFERENCE GREATERP LESSP
                                     PLUS))
               (SPELLINGS3 '(BROKENFNS ADVISEDFNS NOTLISTEDFILES FILELST NOTCOMPILEDFILES PROMPT#FLG
                                     CLISPIFYPRETTYFLG DWIMIFYCOMPFLG FILERDTBL EDITRDTBL SYSPRETTYFLG NOSPELLFLG
                                     INITIALS NIL))
               (SPELLSTR1 "{spellseparator}")
               (SPELLSTR2 "{spellignore}")
               (FIXSPELLREL 70)
               (FIXSPELLDEFAULT '\y)
               (SKORLST1 '(NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL))
               (SKORLST2 '(NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL))
              DWIMKEYLST FIXSPELLKEYLST (FASTYPEFLG)
               (RUNONFLG NIL)
               (\#USERWORDS 20)
               (\#SPELLINGS1 20)
               (\#SPELLINGS2 20)
               (\#SPELLINGS3 20)
               (DWIMWAIT 10)
               (RESPELLS)
               (FIXSPELL.UPPERCASE.QUIET NIL))
        (P (SETSPELLCASE)
           (NCONC1 SPELLINGS1 SPELLSTR1)
           (NCONC1 SPELLINGS2 SPELLSTR1)
           (ATTACH SPELLSTR1 SPELLINGS3))
        (DECLARE\: DONTCOPY (MACROS SPELLEQ))
        (BLOCKS (FIXSPELLBLOCK MISSPELLED? FIXSPELL CHOOZ CHOOZ1 SKOR SKOR0 MOVETOP
                       (ENTRIES MISSPELLED? FIXSPELL CHOOZ SKOR0 SKOR MOVETOP)
                       (LOCALFREEVARS NCXWORD NCTWORD TAIL ALTFLG))
               (FIXSPELL1 FIXSPELL1 FIXSPELL2))
        (DECLARE\: EVAL@COMPILE DONTCOPY (P (AND (OR (GETPROP 'NOSPELLFLG 'GLOBALVAR)
                                                     (FMEMB 'NOSPELLFLG GLOBALVARS))
                                                 (HELP "NOSPELLFLG shouldn't be a global variable!" " How did it
                                                       get that way?"))))
        (GLOBALVARS \#SPELLINGS2 \#SPELLINGS3 \#USERWORDS APPROVEFLG CLISPCHARS CLISPFLG COMMENTFLG DWIMFLG
               DWIMKEYLST DWIMWAIT EDITQUIETFLG FASTYPEFLG LASTWORD REREADFLG RESPELLS RUNONFLG RUNONSTATS
               SKORLST1 SKORLST2 SPELLINGS1 SPELLINGS2 SPELLINGS3 SPELLSTATS SPELLSTATS1 SPELLSTR1 SPELLSTR2
               USERWORDS VETOSTATS)))

(DEFINEQ
```

(ADDSPELL

```
  (LAMBDA (X SPLST N)

          (* |Updates| |appropriate| |spellings| |lists| |as| |follows:| |if| SPLST |is| NIL\, |adds| |to| USERWORDS |and| SPELLINGS2\;
          -
          |if| SPLST |is| 0\, |just| |adds| |to| USERWORDS\; -
          |if| SPLST |is| 1\, |adds| |to| SPELLINGS1\; -
          |if| SPLST |is| 2\, |adds| |to| SPELLINGS2\; -
          |if| SPLST |is| 3\, |adds| |to| USERWORDS |and| SPELLINGS3.
          -
          -
          SPELLINGS1 |is| |the| |list| |of| |functions| |used| |in| |an| APPLY |context,| |e.g.|
          MAKEFILE\, TCOMPL\, DEFINEQ. SPELLINGS2 |is| |the| |list| |of| |functions| |used| |in| |an| EVAL |context,| |e.g.|
          CONS\, AND\, RETURN. SPELLINGS3 |is| \a |list| |of| |variables.|
          USERWORDS |is| \a |list| |of| |both| |functions| |and| |variables| |that| |the| |user| |references.|)

    (AND (LITATOM X)
         (SELECTQ SPLST
              ((NIL 0)
```

(* DEFINE |uses| SPLST NIL\, |adds| |word| |to| SPELLINGS2 |because| |some| |user| |function| |must| |call| |it.|
|However,| |it| |might| |not| |be| \a |top-level| |function,| |so| |it| |isn't| |added| |to| SPELLINGS1 |until| |used| |in| |that| |way.|
-
SPLST 0 |is| |for| LOAD/PROP\, |and| |for| |use| |from| EDITA\, PRINTSTRUCTURE\, |etc.|
|doesn't| |add| |it| |to| SPELLINGS2 |since| |have| |no| |indication| |that| |the| |function| |will| |be| |called| |by| |the| |user.|)

```
        (SETQ USERWORDS (ADDSPELL1 X USERWORDS \#USERWORDS))
        (AND (NULL SPLST)
             (SETQ SPELLINGS2 (ADDSPELL1 X SPELLINGS2 \#SPELLINGS2)))
        (SETQ LASTWORD X))
    (1                                                    (* |Called| |from| LISPX |on| APPLY |inputs,| |add| |to|
                                                          |permanent| |section,| |i.e.| |never| |forgets| X.)
       (SETQ SPELLINGS1 (ADDSPELL1 X SPELLINGS1)))
    (2
```

(* |Called| |from| LISPX |on| EVAL |inputs,| |never| |forgets| X. |Not| |however| |that| |words| |are| |added| |to| |temporary|
|section| |of| SPELLINGS2 |for| TYPE=NIL |or| TYPE=0.)

```
       (SETQ SPELLINGS2 (ADDSPELL1 X SPELLINGS2)))
    ((T 3)
```

(* |Called| |from| LISPX |inputs| |consisting| |of| |just| \a |variable,| |or| |from| SAVESET\, |i.e.|
|any| |call| |to| RPAQ |or| RPAQQ\, |or| |any| |call| |to| SET |or| SETQ |via| LISPX\, |or| |from| EDITV.)

```
        (SETQ USERWORDS (ADDSPELL1 X USERWORDS \#USERWORDS))
        (SETQ SPELLINGS3 (ADDSPELL1 X SPELLINGS3 \#SPELLINGS3))
        (SETQ LASTWORD X))
    (COND
       ((LISTP SPLST)
        (ADDSPELL1 X SPLST N))
       (T (ERROR '"bad addspell type:" SPLST T)))))))
```

(ADDSPELL1

```
(LAMBDA (WORD SPLST N)                                    (* |wt:| " 9-OCT-78 00:37")
```

(* SPLST |is| |divided| |into| |two| |sections,| |permanent| |and| |temporary,| |separated| |by| NIL.
|Words| |are| |never| |forgotten| |from| |the| |permanent| |section.|
A |maximum| |of| N |words| |are| |allowed| |in| |the| |temporary| |section.|
|When| \a |correction| |occurs,| |the| |word| |is| |moved| |to| |the| |front| |of| |the| |list,| |hence| |putting| |it| |in| |the| |permanent|
|section.| -
|When| ADDSPELL1 |is| |called| |with| N=NIL\, |words| |are| |added| |at| |the| |end| |of| |the| |permanent| |section,| |i.e.|
|just| |before| |the| NIL. |Otherwise| |they| |are| |added| |at| |the| |beginning| |of| |the| |temporary| |section,| |and| |if| |there|
|are| |more| |than| N |words| |in| |the| |temporary| |list,| |the| |last| |is| |deleted.|)

```
  (COND
     ((NULL SPLST)
      (SETQ SPLST (LIST SPELLSTR1 WORD)))
     ((AND (NEQ WORD (CAR SPLST))
           (NEQ WORD (CADR SPLST)))                       (* |Loop| |begins| |with| |third| |element.|)
      (PROG ((L1 SPLST)
             (L2 (CDDR SPLST))
             L3 M)
```

(* L1 |stays| |two| |behind| L2 |so| |that| |the| |last| |element| |of| SPLST |can| |be| |deleted.|)

```
            (COND
               ((EQ (CAR SPLST)
                    SPELLSTR1)
                (SETQ L3 SPLST)
                (SETQ M 1)                                (* L2 |has| |already| |skipped| |one| |of| |the| |temporary| |words.|)
                )
               ((EQ (CADR SPLST)
                    SPELLSTR1)
                (SETQ L3 SPLST)
                (SETQ M 0)))
```

(* |Special| |check| |necessary| |because| USERWORDS |starts| |off| |with| NIL |first,| |i.e.|
|no| |permanent| USERWORDS. I\f NIL |not| |noticed,| |length| |won't| |be| |counted| |and| |nothing| |will| |ever| |be| |deleted.|)

```
           LP  (COND
               ((NULL L2)
                (GO OUT))
               ((EQ WORD (CAR L2))
                (COND
                   ((NULL L3)                             (* WORD |is| |already| |in| |permanent| |section.|)
                    )
                   ((NULL N)
```

(* WORD |is| |in| |temporary| |section| |and| |would| |have| |been| |added| |to| |permanent| |section.|
|Add| |it| |to| |permanent| |and| |erase| |from| |temporary.|)

```
                    (FRPLACD (CDR L1)
                             (CDR L2))
                    (FRPLACD L2 (CDR L3))
                    (FRPLACD L3 L2))
```

```
                    (T

          (* WORD |is| |in| |temporary| |section| |and| |would| |have| |been| |added| |to| |temporary| |section.|
          MOVE |it| |to| |front| |of| |temporary| |section.|)

                        (FRPLACD (CDR L1)
                               (CDR L2))
                        (SETQ L3 (CDR L3))
                        (FRPLACD L2 (CDR L3))
                        (FRPLACD L3 L2)))
                   (RETURN))
                  ((EQ (CAR L2)
                       SPELLSTR1)
                   (SETQ L3 (CDR L1))                          (* CAR |of| L3 |is| |the| |last| |member| |of| |the| |permanent|
                                                                |section.|)
                   (SETQ M 0)))
              (SETQ L1 (CDR L1))
              (SETQ L2 (CDR L2))
              (AND M (ADD1VAR M))                              (* M |will| |be| |the| |length| |of| |temporary| |section|)
              (GO LP)
            OUT (COND
                   ((NULL L3)

          (* NIL |not| |found.| |Occurs| |if| |user| |is| |maintaining| |own| |spelling| |list| |and| |not| |using| |temporary/permanent|
          |conventions.|)

                    (NCONC1 L1 WORD))
                   ((NULL N)                                   (* |Add| |at| |end| |of| |permanent| |section.|)
                    (RPLNODE L3 (CAR L3)
                           (CONS WORD (CDR L3))))
                   ((IGREATERP M N)                            (* |Add| |at| |beginning| |of| |temporary| |section,| |delete|
                                                                (AND REUSE) |last| |element| |of| |temporary| |section.|)
                    (RPLNODE (CDR L1)
                           WORD
                           (CDDR L3))
                    (RPLNODE (CDR L3)
                           (CADR L3)
                           (CDR L1))
                    (FRPLACD L1 NIL)                           (* |Not| |worth| |while| |to| |make| |the| |deletion| |of| |last|
                                                                |elemetn| |undoable.|)
                    )
                   (T (RPLNODE (CDR L3)
                             (CADR L3)
                             (CONS WORD (CDDR L3)))            (* |Add| |at| |beginning| |of| |temporary| |section.|)
                     ))
              (AND LISPXHIST (UNDOSAVE (LIST 'ADDSPELL2 WORD SPLST)
                                       LISPXHIST)))))
    SPLST))
```

(ADDSPELL2

```
  (LAMBDA (WORD SPLST)                                         (* |wt:| " 8-OCT-78 23:19")
    (PROG (TEM)
          (AND (SETQ TEM (FMEMB WORD SPLST))
               (/RPLACA TEM SPELLSTR2)))))
```

(MISSPELLED?

```
  (LAMBDA (XWORD REL SPLST FLG TAIL FN)                        (* |wt:| "25-APR-78 12:26")

          (* MISSPELLED? |can| |be| |used| |when| XWORD |may| |in| |fact| |be| |all| |right.|
          FIXSPELL |should| |be| |used| |if| |you| |know| XWORD |is| |wrong.|)

    (PROG NIL                                                  (* REL |is| |between| 0 |and| 100 |and| |indicates| |percentage.|)
          (RETURN (COND
                     ((OR (NULL XWORD)
                          (EQ XWORD 'Ÿ))
                      (LISPXWATCH SPELLSTATS1)
                      (PRIN1 '= T)
                      (PRINT LASTWORD T T))
                     ((COND
                         ((NULL FN)
                          (FMEMB XWORD SPLST))
                         (T (APPLY* FN XWORD)))
                      XWORD)
                     (T (FIXSPELL XWORD REL SPLST FLG TAIL FN)))))))
```

(FIXSPELL

```
  (LAMBDA (XWORD REL SPLST FLG TAIL FN TIEFLG DONTMOVETOPFLG FROMDWIM APPROVALFLG)
                                                               (* |lmm| "15-Nov-86 22:22")
```

;; If FLG is T, XWORD is printed to left of = sign.

;; CLST is used when FIXSPELL is called from WTFIX. In this case, if TYPE-IN? is NIL, XWORD is printed, and a -> is used instead of =. In ;; addition, if FAULTFN is not NIL, (IN FAULTFN) is also printed as part of the message.

;; If CLST is a list, it is a DUNPACK of XWORD, since in most cases WTFIX will already have computed this list.

```
;;  FLG is used to specify other types of messages besides ->.

;; If TAIL is supplied, and word is equal to CAR of it, the correction will be smahed into TAIL.  If TAIL is non NIL, runon corrections will be
;; attempted.  (If TAIL=T, and a runon corection is approved, the dotted pair is returned as the value.)

;; IF FLG=NO-MESSAGE, the correction is returned without asking for approval.

(PROG (X FIXSPELLTEM (NDBLS 0)
         TLST
         (DWIM.GIVE.UP.TIME (OR DWIM.GIVE.UP.TIME (SETUPTIMER DWIM.GIVE.UP.INTERVAL))))
       (COND
          ((OR (EQ NOSPELLFLG T)
               (AND NOSPELLFLG FROMDWIM (NOT TYPE-IN?))
               (AND (NLISTP SPLST)
                    (NOT (ARRAYP SPLST)))
               (AND (NOT (LITATOM XWORD))
                    (NOT (STRINGP XWORD))))
           (RETURN)))
       (COND
          ((NULL REL)
           (SETQ REL FIXSPELLREL)))
       (COND
          ((NULL XWORD)
           (RETURN NIL)))
       (SETQ TLST (CHCON XWORD))
       (COND
          ((AND (LISTP SPLST)
                (NOT (STACKP (CAR SPLST))))
           (|for| Z |in| SPLST |when| (LISTP Z) |do| (COND
                                                       ((EQ (CAR Z)
                                                            XWORD)
                                                        (SETQ X (CDR Z))
                                                        (COND
                                                           (FROMDWIM (GO FIXSPELLOUT))
                                                           (T (GO FIXSPELLRET))))))
           (COND
              ((NOT (U-CASEP XWORD))
               (SETQ FIXSPELLTEM (U-CASE XWORD))
               (COND
                  ((|if| FN
                         |then| (CL:FUNCALL FN FIXSPELLTEM)
                     |else| (OR (COND
                                   ((OR (EQ SPLST SPELLINGS1)
                                        (EQ SPLST SPELLINGS2))
                                    (CL:FBOUNDP FIXSPELLTEM))
                                   ((EQ SPLST SPELLINGS3)
                                    (BOUNDP FIXSPELLTEM)))
                                (FMEMB FIXSPELLTEM SPLST)))
                   (SETQ X FIXSPELLTEM)
                   (COND
                      ((AND FIXSPELL.UPPERCASE.QUIET (OR (NOT FROMDWIM)
                                                         (EQ TYPE-IN? T)))
                       (GO FIXSPELLRET))
                      (T (GO FIXSPELLOUT))))
                  ((SOME SPLST (FUNCTION (LAMBDA (X)
                                         (AND (LISTP X)
                                              (EQ FIXSPELLTEM (CAR X))
                                              (SETQ FIXSPELLTEM X)))))

                   ;; A synonym on the spelling list.

                   (SETQ X (LIST (CAR FIXSPELLTEM)
                                 (CDR FIXSPELLTEM)))
                   (GO LP2)))))))
       (COND
          ((AND (EQ XWORD 'Ÿ)
                (OR (NULL FROMDWIM)
                    TYPE-IN?))                                  ; TYPE-IN?  is bound in WTFIX.
           (SETQ X LASTWORD)
           (|FIXSPELL1| XWORD LASTWORD NIL FROMDWIM)
           (GO FIXSPELLRET))
          ((AND (SETQ FIXSPELLTEM (FASSOC XWORD (LISTGET1 LISPXHIST 'RESPELLS)))
                (FMEMB (CDR FIXSPELLTEM)
                       SPLST))                                  ; Already made this correctionthis event.
           (SETQQ APPROVALFLG NEEDNOTAPPROVE)
           (SETQ X (CDR FIXSPELLTEM))
           (GO LP2)))
       (SETQ X TLST)
  LP   (COND
          ((NULL X)
           (GO LP1))
          ((AND (EQ (CAR X)
                    (CHARCODE ESC))
                (SETQ FIXSPELLTEM (CDR X)))

           ;; for escape codes we call CHOOZ since this also handles the case where ther are misspellings in the leading characters.

           (SETQ TLST (UNPACK XWORD))
           (PROGN                                               ; Alt-mode matching.
                  (RESETVARS ((EDITQUIETFLG T))
                             (PROG ((L SPLST)
```

```
                                    (GENFN (AND (ARRAYP SPLST)
                                                (ELT SPLST 1)))
                                    (GENERATOR (AND (STACKP (CAR SPLST))
                                                    SPLST)))
                                (SETQ X NIL)
                             LP3 (COND
                                    (GENFN (COND
                                              ((NULL (SETQ FIXSPELLTEM (APPLY* GENFN SPLST)))
                                               (RETURN))))
                                    (GENERATOR (COND
                                                  ((EQ (SETQ FIXSPELLTEM (GENERATE GENERATOR))
                                                       GENERATOR)
                                                   (RELSTK (CDR GENERATOR))
                                                   (RETURN))))
                                    ((NULL L)
                                     (RETURN))
                                    ((NULL (SETQ FIXSPELLTEM (CAR L)))
                                     (SETQ L (CDR L))
                                     (GO LP3))
                                    (T (SETQ L (CDR L))))
                                 (COND
                                    ((AND (EDIT4E1 TLST (UNPACK FIXSPELLTEM))
                                          (OR (NULL FN)
                                              (CL:FUNCALL FN FIXSPELLTEM)))
                                     (AND GENFN (STRINGP FIXSPELLTEM)
                                          (SETQ FIXSPELLTEM (MKATOM FIXSPELLTEM)))
                                                                  ; because the generator function may (frequently does) reuse the
                                                                  ; string it reutnrs.
                                     (COND
                                        ((OR (EQ TIEFLG 'ALL)
                                             (EQ TIEFLG 'LIST)
                                             (EQ TIEFLG 'EVERYTHING))
                                         (SETQ X (CONS FIXSPELLTEM X)))
                                        (TIEFLG (SETQ X FIXSPELLTEM))
                                        (X                        ; Already a match, therefore ambiguous.
                                           (PRINT '|ambiguous| T)
                                           (SETQ X NIL)
                                           (RETURN))
                                        (T (SETQ X FIXSPELLTEM)))))
                                 (GO LP3)))
                      (GO LP2)))
              ((EQ (CAR X)
                   FIXSPELLTEM)
               (SETQ NDBLS (ADD1 NDBLS)))
              (T (SETQ FIXSPELLTEM (CAR X))                       ; FIXSPELLTEM keeps track of the previous character.

                 ))
          (SETQ X (CDR X))
          (GO LP)
       LP1 (SETQ X (CHOOZ TLST REL SPLST TAIL FN TIEFLG NDBLS FROMDWIM))
       LP2 (COND
              ((NULL X)
               (RETURN NIL))
              ((OR (EQ TIEFLG 'ALL)
                   (EQ TIEFLG 'LIST)
                   (EQ TIEFLG 'EVERYTHING))
               (RETURN X))
              ((LISTP X)
               (RETURN (COND
                          ((LISTP (CDR X))                        ; synonym correction.  XWORD is identical with CAR of X.
                           (COND
                              ((OR (EQ XWORD (CAR X))
                                   (EQ FLG 'NO-MESSAGE))          ; no approval necessary
                               (SETQ X (CADR X))
                               (GO FIXSPELLRET))
                              ((SETQ FIXSPELLTEM (FIXSPELL1 XWORD (CAR X)
                                                            FLG FROMDWIM APPROVALFLG))
                                                                  ; e.g.  synonym is S.T.  but XWORD is S.TT.
                               (SETQ X (COND
                                          ((LISTP FIXSPELLTEM)    ; user specified new value via USING
                                           (CAR FIXSPELLTEM))
                                          (T (CADR X))))
                               (GO FIXSPELLRET))))
                          ((NULL TAIL)                            ; value of form (a  . b) returned by chooz means runon correction
                           NIL)
                          ((EQ FLG 'NO-MESSAGE)
                           X)
                          ((SETQ FIXSPELLTEM (FIXSPELL1 XWORD (COND
                                                                ((LISTP (CAR X))
                                                                 ;; both a runon and synonym involved, e.g.  user types WHERE, and (WHE  . SY) on
                                                                 ;; spelling list.  fixpsspell1 asks WHERE= WHE RE?
                                                                  (CONS (CAAR X)
                                                                        (CDR X)))
                                                                (T X))
                                                             FLG FROMDWIM (OR APPROVALFLG 'MUSTAPPROVE)))
                                                                  ; Runon correction.
```

```
                        (COND
                           ((LISTP FIXSPELLTEM)                ; user typed in a value.  no run on
                            (SETQ X (CAR FIXSPELLTEM))
                            (GO FIXSPELLRET))
                           ((AND (LISTP TAIL)
                                 (EQ XWORD (CAR TAIL)))        ; Smash TAIL and eturn the first word
                            (/RPLNODE TAIL (COND
                                              ((LISTP (CAR X))
                                               (CADAR X))
                                              (T (CAR X)))
                                   (CONS (CDR X)
                                         (CDR TAIL)))
                            (CAR TAIL))
                           ((LISTP FIXSPELLTEM)
                            (COND
                               ((CDR FIXSPELLTEM)
                                (CONS (CAR FIXSPELLTEM)
                                      (CADR FIXSPELLTEM)))
                               (T (CAR FIXSPELLTEM))))
                           (T                                  ; Return the dotted pair
                              X)))
                     ((IGREATERP (CHOOZ1 (NCHARS XWORD)
                                         (NCHARS (COND
                                                    ((LISTP (CAR X))
                                                     (CAAR X))
                                                    (T (CAR X))))
                                         (NCHARS (CDR X)))
                           REL)
                      ;; e.g.  the correction (BREAK  . X) is offered for BREAKX this asks whether BREAK is a valid correction for BREAKX
                      (SETQ X (CAR X))
                      (GO LP2))))))
  FIXSPELLOUT
      (COND
         ((EQ FLG 'NO-MESSAGE)
          (RETURN X))
         ((SETQ FIXSPELLTEM (FIXSPELL1 XWORD X FLG FROMDWIM APPROVALFLG))
                                                             ; Prints appropriate message to user.
          (COND
             ((LISTP FIXSPELLTEM)
              (SETQ X (CAR FIXSPELLTEM))
              (ADDSPELL X SPLST))
             ((AND (NULL DONTMOVETOPFLG)
                   (LISTP SPLST)
                   (FMEMB SPELLSTR1 SPLST))
              (MOVETOP X SPLST)))
          (GO FIXSPELLRET))
         (T (RETURN NIL)))
  FIXSPELLRET
      (PROGN (AND (LISTP TAIL)
                  (EQ XWORD (CAR TAIL))
                  (/RPLNODE TAIL X (CDR TAIL)))
             (RETURN X))))
```

(FIXSPELL1

```
  (LAMBDA (WORD X FLG FROMDWIM APPROVALFLG DEFAULT)            (* |lmm| "15-Nov-86 22:22")
                                                               ; Performs interaction with user associated with a spelling or
                                                               ; other correction.
    (PROG ((LISPXHIST (AND LISPXHIST (CONS '*LISPXPRINT* (CONS NIL LISPXHIST))))
           (VAL T)
           MESSFLG BUFS TEM)
      ;; LISPXHIST is rebound as we don't want to include any DWIM messages if the change isn't actually performed, e.g.  the user says NO.
      ;; Therefore, we want to have LISPXPRIN1 et al save the output on a local LISPXHIST, and then transfer it all over to the real LISPXHIST if the
      ;; correction goes through.
          (AND (EQ WORD X)
               (ERROR!))
          (COND
             ((NEQ (POSITION T)
                   0)
              (LISPXTERPRI T)
              (TERPRI T)))
          (COND
             ((OR (SETQ MESSFLG (OR (STRINGP WORD)
                                    (STRINGP X)))
                  (EQ RESPELLS T))                             ; Dont keep respells.

              )
             ((OR REREADFLG (STRPOS '"Ÿ" WORD))                ; Spelling completion

              )
             ((OR (LISTP X)
                  (NULL (SETQ TEM (FASSOC X RESPELLS))))
              (SETQ RESPELLS (CONS (LIST X WORD)
                                   RESPELLS)))
             (T (NCONC1 TEM WORD)))
```

```
          (COND
             ((AND (NEQ APPROVALFLG 'MUSTAPPROVE)
                   (COND
                      (FROMDWIM (AND TYPE-IN? (NULL MESSFLG)))
                      (T (NULL FLG))))
```

;; This is the case where the correction is a spelling correction (as inidcated by MESSFLG being NIL), and no approval is needed. i.e. ;; FIXSPELL1 is just going to print = followed by the word.

```
              (AND (OR (EQ REREADFLG 'T)
                       FLG)
                   (PRIN2 WORD T T))
              (PRIN1 "=" T)
              (FIXSPELL2 X)
              (GO OUT1)))
          (COND
             ((AND (NEQ APPROVALFLG 'MUSTAPPROVE)
                   (COND
                      ((NULL FROMDWIM)
                       (OR (NULL FLG)
                           (NULL APPROVEFLG)))
                      (T
```

;; OR is true if approval is required. NOte that even if APPROVEFLG is T, when there are two interpretations to a ;; correction, as indicated by CLISPCHANGES not being NIL, always aks approval.

```
                         (NULL (OR (AND (NULL TYPE-IN?)
                                        APPROVEFLG)
                                   CLISPCHANGES)))))
              (SETQQ APPROVALFLG NEEDNOTAPPROVE)))
          (COND
             ((OR (EQ APPROVALFLG 'MUSTAPPROVE)
                  (AND (NEQ APPROVALFLG 'NEEDNOTAPPROVE)
                       APPROVEFLG))
```

;; Want to clear out LINUF and SYSBUF to prevent CLBUFS from misakenly returning left over typeahead from a previous ;; CLEARBUF.

```
              (LINBUF)
              (SYSBUF)                                  ; The extra argument to CLBUFS prevents READBUF (LISPX'S
                                                        ; buffer) from being cleared.
              (SETQ BUFS (CLBUFS NIL T READBUF))))
          (FIXSPELL2 WORD T)
          (COND
             ((AND FROMDWIM (NULL TYPE-IN?))
              (FIXPRINTIN FAULTFN T)
              (LISPXPRIN1 (OR FLG '" -> ")
                     T))
             (T (LISPXPRIN1 (COND
                               ((AND FLG (NEQ FLG T))
                                FLG)
                               (T '=))
                            T)                          ; E.g. For Shall I load ... message, FLG is '' for unary minus it is
                                                        ; ''

           ))
          (AND NIL (STRINGP WORD)
               (STRINGP X)
               (OR (NULL FLG)
                   (EQ FLG T))
               (NOT (STREQUAL WORD '""))
               (LISPXTERPRI T))
```

;; On corrections where both left and right are strings, and FLG is normal (thereby xcluding the TREAT AS CLISP case) print the strings on ;; separate lnes for readability.

```
          (COND
             ((EQ APPROVALFLG 'NEEDNOTAPPROVE)
              (FIXSPELL2 X)
              (GO OUT)))
          (FIXSPELL2 X T)
          (SETQ VAL (ASKUSER (AND DWIMWAIT (COND
                                              (MESSFLG
```

;; MESSFLG would be NIL for straight spelling correction. This says that the correction involves an 8 ;; or a 9, or asks some question about CLISP. User will probably need more time to think about it in ;; this case.

```
                                                       (ITIMES 3 DWIMWAIT))
                                              (T DWIMWAIT)))
                             (COND
                                (DEFAULT)
                                ((AND (LISTP X)
                                      (OR (ILESSP (SETQ TEM (NCHARS (CAR X)))
                                                  3)
                                          (NOT (IGREATERP TEM (NCHARS (CDR X))))))
```

;; Runon correction. Defaut is NO if less than three characters in first word, or first word is not greater than ;; second in length

```
                                 '\n)
                                (T FIXSPELLDEFAULT))
                             '" ?  "
                             (COND
```

```
                              (FROMDWIM DWIMKEYLST)
                              (T FIXSPELLKEYLST))))
          (AND BUFS (BKBUFS BUFS))
          (SELECTQ VAL
               ((Y \y)
                    (SETQ VAL T))
               (N (LISPXWATCH VETOSTATS)                    ; If value returned was 'n' as opposed to 'N', means defaulted to
                                                            ; NO, not vetoed.
                  (RETURN NIL))
               (\n (RETURN NIL))
               (SETQ X (CAR VAL)))
          (LISPXTERPRI T NIL NIL T)                         ; Adds a TERPRI to history list if LISPXPRINTFLG is T, but does
                                                            ; not actually print a carriage return.
      OUT (COND
             ((CADR LISPXHIST)
              (LISPXPUT '*LISPXPRINT* (CADR LISPXHIST)
                     T
                     (CDDR LISPXHIST))
              ;; Makes the print information part of LISPXHIST.  Before it was on a property that ws just consed onto the front.  This will also add it
              ;; to any other print information.

              ))
      OUT1
          (AND FROMDWIM LISPXHIST (NULL TYPE-IN?)
               (NEQ (CAR SIDES)
                    'CLISP\ )
               (SETQ SIDES (LIST 'CLISP\  (LIST COMMENTFLG (CADR LISPXHIST)
                                                SIDES))))
    ;; This marks the side information and print information as of the beginning of this correction.  For usefor selective undoing.  CADR of LISPXHIST
    ;; (which was rebound here), will be the beginning of the PRINT information, which if approved, will be NCONCed onto the print informaion for this
    ;; event.
          (COND
             (MESSFLG (RETURN VAL))
             (FROMDWIM (LISPXWATCH SPELLSTATS)
                    (AND LISPXHIST (LISPXPUT 'RESPELLS (LIST (CONS WORD X))
                                             T LISPXHIST)))
             (T (LISPXWATCH SPELLSTATS1)))
          (AND (LISTP X)
               (LISPXWATCH RUNONSTATS))
          (RETURN VAL))))
```

(**FIXSPELL2**

```
(LAMBDA (X FLG)                                             (* |wt:| 15-JUL-76 20 53)
  (COND
     ((LISTP X)
      (MAPRINT (COND
                  ((AND (CDR X)
                        (NLISTP (CDR X)))
                   (LIST (CAR X)
                         (CDR X)))
                  (T X))
               T NIL NIL NIL (FUNCTION (LAMBDA (X)
                                       (COND
                                          ((STRINGP X)
                                           (LISPXPRIN1 X T))
                                          (T (LISPXPRIN2 X T T)))))
               T))
      ((STRINGP X)
       (LISPXPRIN1 X T))
      (T (LISPXPRIN2 X T T)))
  (COND
     ((NULL FLG)
      (LISPXTERPRI T)))))
```

(**CHOOZ**

```
(LAMBDA (XWORD REL SPLST TAIL FN TIEFLG NDBLS FROMDWIM)     (* |lmm| "15-Nov-86 22:12")
  (COND
     ((NLISTP XWORD)                                        ; When called from FIXSPELL, XWORD is already a CHCON lst.
      (SETQ XWORD (CHCON XWORD))))
  (PROG ((NCXWORD0 (FLENGTH XWORD))
         NCXWORD NCTWORD TWORD TWORD1 TWORD2 TEM SC VAL (GENFN (AND (ARRAYP SPLST)
                                                                    (ELT SPLST 1)))
         (GENERATOR (AND (STACKP (CAR SPLST))
                         SPLST))
         ALTFLG)
        (COND
           ((NULL NDBLS)
            (SETQ NDBLS 0)
            (MAPC XWORD (FUNCTION (LAMBDA (X)
                                  (COND
                                     ((EQ X TEM)
                                      (SETQ NDBLS (ADD1 NDBLS)))
                                     (T (SETQ TEM X))))))))  ; Counts number of (possibly) doubled characters
        (SETQ ALTFLG (EQ (CAR (LAST XWORD))
```

```
                        (CHARCODE ESC)))                        ; xword ends in an alt-mode.  means k to call skor even if
                                                                ; testword is much longer.
      LP  (AND DWIM.GIVE.UP.TIME (TIMEREXPIRED? DWIM.GIVE.UP.TIME)
               (GO OUT))
          (COND
             (GENFN ;; this provides a way of giving the spelling corrector a generating function intead of a spelling list.  the generatng function can
                    ;; keep its 'state' in one of the other cells of the array.  when it returns a value of NIL for the 'next' element, the spelling lists is
                    ;; assume xhausted.
                    (COND
                       ((NULL (SETQ TWORD (APPLY* GENFN SPLST)))
                        (GO OUT))))
             (GENERATOR (COND
                           ((EQ (SETQ TWORD (GENERATE GENERATOR))
                                GENERATOR)
                            (RELSTK (CDR GENERATOR))
                            (GO OUT))))
             ((NULL SPLST)
              (GO OUT))
             ((OR (EQ (SETQ TWORD (CAR SPLST))
                      SPELLSTR1)
                  (EQ TWORD SPELLSTR2))                         ; marker.
              (SETQ SPLST (CDR SPLST))
              (GO LP))
             (T (SETQ SPLST (CDR SPLST))))
          (COND
             ((LISTP TWORD)                                     ; Synonym feature.
              (SETQ TWORD1 (CAR TWORD))
              (SETQ TWORD2 (CDR TWORD)))
             (T (SETQ TWORD1 (SETQ TWORD2 TWORD))))
          (SETQ NCTWORD (NCHARS TWORD1))
          (SETQ NCXWORD (COND
                           (ALTFLG
                                  ;; for purposes of call to skor, pretend that both words are same length so first character matched against
                                  ;; first character.
                                  NCTWORD)
                           (T NCXWORD0)))
          (COND
             ((COND
                 ((IGREATERP NCTWORD NCXWORD)
                  ;; Checks to see if test word and unknown word differ sufficiently in number of characters so as to make it unnecessary to even
                  ;; call SKOR.  This case is where test word is longer than XWORD.  If number of characters in XWORD, NCW, divided by
                  ;; number of characters in test word, NCT, is less than REL than don't bother to call SKOR.  0 P
                  (AND (NULL ALTFLG)
                       (ILESSP (IQUOTIENT (ITIMES NCXWORD 100)
                                          NCTWORD)
                               REL)))
                 ((ILESSP (IQUOTIENT (ITIMES NCTWORD 100)
                                     (ADD1 (IDIFFERENCE NCXWORD NDBLS)))
                          REL)                                  ; XWORD longer than test word.  However, must allow for
                                                                ; possibility of doubled characters.
                  T))
              (GO LP))
             ((AND (SETQ SC (SKOR XWORD (SETQ TEM (DCHCON TWORD1 SKORLST2))
                                  NCXWORD NCTWORD FROMDWIM))
                   (OR (NULL FN)
                       (CL:FUNCALL FN TWORD2)))
              (SETQ TEM (COND
                           ((LISTP TWORD)                       ; to distinguish from a runon correction, which is returned as a
                                                                ; dotted pair.
                            (LIST TWORD1 TWORD2))
                           ((AND GENFN (STRINGP TWORD))
                            (MKATOM TWORD))
                           (T TWORD)))                          ; note that i dont know what happens if you have both a synonym
                                                                ; and runoncorrecton
              (COND
                 ((LISTP SC)
                  (AND RUNONFLG TAIL (OR (NULL VAL)
                                         (EQ TIEFLG 'EVERYTHING)
                                         (IGREATERP NCTWORD (NCHARS (CAAR VAL))))
                       (SETQ VAL (CONS (CONS TEM (PACKC SC))
                                       (COND
                                          ((EQ TIEFLG 'EVERYTHING)
                                           VAL)))))
                  ;; TWORD1 used instead of TWORD2 becauseif any interaction, want user to approve in terms of his typeing, not the synonym.
                  ;; this will mean another call to spelling corrector to get the synonym, but big deal.

                  )
                 ((ZEROP SC)
                  (COND
                     ((EQ TIEFLG 'EVERYTHING)
                      (SETQ VAL (CONS TEM VAL)))
                     ((AND (NEQ TIEFLG 'ALL)
                           (NEQ TIEFLG 'LIST))                  ; return the value
                      (SETQ VAL TEM)
```

```
                    (GO OUT1))
                   ((NEQ REL 100)                          ; tieflg=LIST means list the tied candidates.  it used to be called
                                                           ; ALL
                    (SETQ REL 100)
                    (SETQ VAL (LIST TEM)))
                   (T (SETQ VAL (CONS TEM VAL)))))
             ((IGREATERP (SETQ SC (COND
                                  (ALTFLG (CHOOZ1 (SUB1 (IDIFFERENCE NCXWORD0 (IDIFFERENCE NCTWORD
                                                                                    NCXWORD)))
                                                  (SUB1 NCXWORD0)
                                                  SC))
                                  (T (CHOOZ1 NCXWORD NCTWORD SC))))
                         REL)
              (SETQ VAL (CONS TEM (COND
                                  ((EQ TIEFLG 'EVERYTHING)
                                   VAL)
                                  (T (SETQ REL SC)         ; Now only look for words CLOSER than SC.
                                     NIL)))))
             ((EQ SC REL)
              (SETQ VAL (CONS TEM VAL))))))
        (GO LP)
    OUT (SETQ VAL (COND
                  ((OR (EQ TIEFLG 'ALL)
                       (EQ TIEFLG 'LIST)
                       (EQ TIEFLG 'EVERYTHING))
                   (COND
                      ((CDR VAL)
                       (DREVERSE VAL))
                      (T VAL)))
                  ((AND (CDR VAL)
                        (NULL TIEFLG))                     ; More than one.
                   NIL)
                  (T (CAR VAL))))
    OUT1
        (RETURN VAL))))
```

(CHOOZ1

```
(LAMBDA (NC1 NC2 SC)                                       (* |wt:| 29-NOV-76 14 53)
  (PROG (TEM)

        (* |The| |arithmetic| |expression| |computes| |the| |relative| |closeness| |as| \a |percentage|
        (|times| 100) |by| |dividing| |the| |difference| |between| |the| |average| |number| |of| |characters| |and| |the| |number| |of|
        |mistakes,| |over| |the| |average| |number| |of| |characters.| |This| |is|
        ((((|a+b|) /2) -
        |sc|) / (|a+b|) /2 |Multiplying| |top| |and| |bottom| |by| |two| |gives|
        (A+B-2*SC/A+B))

        (RETURN (IQUOTIENT (ITIMES 100 (IDIFFERENCE (SETQ TEM (IPLUS NC1 NC2))
                                                    (ITIMES SC 2)))
                           TEM)))))
```

(SETSPELLCASE

```
(LAMBDA NIL                                                (* |lmm| " 1-JUN-84 23:44")
  (SETQ SPELLCASEARRAY (CASEARRAY))
  (FOR I FROM (CHARCODE A) TO (CHARCODE Z) DO (SETCASEARRAY SPELLCASEARRAY (IPLUS I (IDIFFERENCE
                                                                                     (CHARCODE \a)
                                                                                     (CHARCODE A)))
                                                                                  I))
  (FOR X IN '((1 !)
              (2 \")
              (3 \#)
              (4 $)
              (5 %)
              (6 &)
              (7 \' &)
              (8 \( *)
              (9 \) \()
              (0 \) _)
              (= - +)
              (\; +)
              (\' \")
              (\: * \;)
              (< \,)
              (> \.)
              (? /))
    DO (FOR Y IN (CDR X) DO (SETCASEARRAY SPELLCASEARRAY (CHCON1 Y)
                                          (CHCON1 (CAR X)))))))
```

(SKOR0

```
(LAMBDA (TWORD NCXWORD NDBLS LST)                          (* |bvm:| " 4-Nov-86 01:56")

        (* A |special| |call| |to| SKOR |for| |use| |by| |editor.| LST |is| |an| |exploded| |chconlst| |of| |characteers,| NCXWORD |the|
        |number| |of| |characters| |in| L\, NDBLS |the| |number| |of| |doubled| |characters.|
        SKOR0 |compares| TWORD |with| L\, |and| |returns| T |if| |'close'.|)
```

```
    (PROG ((NCTWORD (NCHARS TWORD))
           SC TEM TAIL ALTFLG)
          (RETURN (AND (COND
                          ((IGREATERP NCTWORD NCXWORD)
                           (NOT (ILESSP (IQUOTIENT (ITIMES NCXWORD 100)
                                                   NCTWORD)
                                        70)))
                          (T (IGREATERP (IQUOTIENT (ITIMES NCTWORD 100)
                                                   (COND
                                                      ((EQ NCXWORD NDBLS)
                                                       1)
                                                      (T (IDIFFERENCE NCXWORD NDBLS))))
                                        70)))
                       (NUMBERP (SETQ SC (SKOR LST (CHCON TWORD)
                                                   NCXWORD NCTWORD)))
                       (OR (ZEROP SC)
                           (IGREATERP (IQUOTIENT (ITIMES (IDIFFERENCE (SETQ TEM (COND
                                                                                  ((IGREATERP NCXWORD NCTWORD)
                                                                                   NCXWORD)
                                                                                  (T NCTWORD)))
                                                                      SC)
                                                         100)
                                                 TEM)
                                      70)))))
```

(SKOR

```
(LAMBDA (XWORD TWORD NCX NCT FROMDWIM)                    (* |lmm| "15-Nov-86 22:28")
```

;; This algorithm counts the number of mistakes in the testword vis a vis the known word. A mistake is a character in the known word that does not ;; have a corresponding character in the test word, or vice versa. Mistakes are not counted until the end of the scoring, so that transpositions are ;; not counted as mistakes. Instead, whenever an unexplained character is encountered, the tail is put in a buffer for the corresponding word. (For ;; reasons of efficiency, instead of a genuine buffer, two PROG variables are used for each word: T1, T2, X1, and X2. Whenever these 'buffers' are ;; exceeded, the skoring is aborted and NIL is returned as the value of SKOR.) When a character is found that does not match, it is first compared ;; with the buffer for the other word. If it is there, it is not counted as a mistake but as out of order. Out of order characters are counted as mistakes ;; if they are misplaced by more than two positions, or if there are any other mistakes, e.g. substitutions or missing letters. Also, double letters are ;; not counted as mistakes, nor are shift mistakes, e.g. @RINT vs PRINT gives a value of 0

```
(PROG (X1 X2 T1 T2 X-1 XC TC (N 0)
          (NTRANS 0)
          TEM)
  LP   (SETQ XC (CAR XWORD))
       (SETQ TC (CAR TWORD))
       (COND
          ((NULL XWORD)
           (COND
              ((NULL TWORD)
               (GO OUT))
              (T (GO LP2))))
          ((EQ XC 27)                                     ; altmode
           (COND
              ((SETQ XWORD (CDR XWORD))
               (RETURN)))
           (SETQ TWORD NIL)
           (GO LP1))
          ((NULL TWORD)
           (GO LP1))
          ((SPELLEQ XC TC)
           (SETQ XWORD (CDR XWORD))
           (SUB1VAR NCX)
           (SETQ TWORD (CDR TWORD))
           (SUB1VAR NCT)
           (SETQ X-1 XC)
           (GO LP)))
  LP1 (COND
         ((AND T2 (SPELLEQ XC (CAR T2)))
```

;; Character encountered in TWORD before XWORD, e.g. the P in IPRNT vs PRINT. The case of RPINT vs PRINT is handled ;; specially without ever going to the buffers.

```
          (COND
             ((IGREATERP (FLENGTH T2)
                         (IPLUS NCX 2))
              (ADD1VAR N))
             (T (ADD1VAR NTRANS)))
          (SETQ T2 NIL)
          (SETQ XWORD (CDR XWORD))
          (SUB1VAR NCX)
          (SETQ X-1 XC)
          (GO LP))
         ((AND T1 (SPELLEQ XC (CAR T1)))
          (COND
             ((IGREATERP (FLENGTH T1)
                         (IPLUS NCX 2))
              (ADD1VAR N))
             (T (ADD1VAR NTRANS)))
          (COND
             (T2 (SETQ T1 T2)
```

```
                    (SETQ T2 NIL))
                   (T (SETQ T1 NIL)))
             (SETQ XWORD (CDR XWORD))
             (SUB1VAR NCX)
             (SETQ X-1 XC)
             (GO LP))
            ((NULL TWORD)
             (GO LP3)))
      LP2 (COND
            ((AND X2 (SPELLEQ TC (CAR X2)))
             (COND
                ((IGREATERP (FLENGTH X2)
                         (IPLUS NCT 2))
                 (ADD1VAR N))
                (T (ADD1VAR NTRANS)))                          ; Character encountered in XWORD first, e.g.  I in IPRNT vs
                                                               ; PRINT.
             (SETQ X2 NIL)
             (SETQ TWORD (CDR TWORD))
             (SUB1VAR NCT)
             (GO LP))
            ((AND X1 (SPELLEQ TC (CAR X1)))
             (COND
                ((IGREATERP (FLENGTH X1)
                         (IPLUS NCT 2))
                 (ADD1VAR N))
                (T (ADD1VAR NTRANS)))
             (COND
                (X2 (SETQ X1 X2)
                    (SETQ X2 NIL))
                (T (SETQ X1 NIL)))
             (SETQ TWORD (CDR TWORD))
             (SUB1VAR NCT)
             (GO LP))
            ((AND XWORD (EQ XC (CADR TWORD))
                  (EQ TC (CADR XWORD))
                  (NEQ TC (CADDR TWORD)))
             ;; Special check for most common case of transposition.  The last clause is an attempt to distinguish the case of a transposition from
             ;; simply getting out of synch.  e.g.  consider MYCIN vs MICIN.  The Y is discarded, and then we are comparing CIN with ICIN.
             ;; Treating CI as a transposition of IC is wrong in this case, since it matches with CI if the I is discarded.
             (SETQ X-1 (CADR XWORD))
             (SETQ XWORD (CDDR XWORD))
             (SUB1VAR NCX)
             (SUB1VAR NCX)
             (ADD1VAR NTRANS)
             (SETQ TWORD (CDDR TWORD))
             (SUB1VAR NCT)
             (SUB1VAR NCT)
             (GO LP))
            ((IGREATERP NCT NCX)                               ; Remove from TWORD.
             (COND
                ((NULL T1)
                 (SETQ T1 TWORD))
                ((NULL T2)
                 (SETQ T2 TWORD))
                ((AND ALTFLG (OR (EQ XC X-1)
                                 (EQ XC (CADR XWORD))))
                 ;; we already have two unaccounted for characters in tword, the (still) longer word.  no point in checking for doubled character in
                 ;; xword, because even if it were, still would be three characters unaccounted for.  however, if altflg is T, then worthwhile.
                 ;; reason why we dont do this before goig through T1 and T2 is that it might NOT be a doubled character, but a missplaced
                 ;; character.
                 (GO LP3))
                (T (RETURN NIL)))
             (SETQ TWORD (CDR TWORD))
             (SUB1VAR NCT)
             (GO LP)))
      LP3 (COND
            ((OR (EQ XC X-1)
                 (EQ XC (CADR XWORD)))
             ;; About to remove from XWORD, check for double char.  first check says was equal to last character.  This occurs when last
             ;; character was correct.  Second check says equal to next character, so throw this one away.
             (SETQ XWORD (CDR XWORD))
             (SETQ NCXWORD (SUB1 NCXWORD))
             ;; Bound in CHOOZ.  When computing value of SKOR, want to divide number of mistakes by actual length of word, i.e.  length minus
             ;; number of doubled characters.  Otherwise, making a word longer by adding extra characters will make it CLOSER, e.g.  ZZZZZZZ
             ;; would correct to PP.
             (SUB1VAR NCX))
            (T (COND
                  ((NULL X1)
                   (SETQ X1 XWORD))
                  ((NULL X2)
                   (SETQ X2 XWORD))
                  (T (RETURN NIL)))
               (SETQ XWORD (CDR XWORD))
```

```
               (SUB1VAR NCX)
               (SETQ X-1 XC)))
         (GO LP)
      OUT (COND
            ((AND (NULL XWORD)
                  (NULL TWORD)
                 T1 X1)
             (SETQ T1 (FLENGTH T1))
             (SETQ X1 (FLENGTH X1))
             (AND T2 (SETQ T2 (FLENGTH T2)))
             (AND X2 (SETQ X2 (FLENGTH X2)))
             (COND
                ((OR (EQ T1 X1)
                     (EQ T1 X2))
                 ;; Check for substitution errors.  Subtracts one so when two gets added below, net effect is only counted as one.
                 (SUB1VAR N)))
             (COND
                ((AND T2 (OR (EQ T2 X1)
                             (EQ T2 X2)))
                 (SUB1VAR N)))))
         (SETQ N (IPLUS N (COND
                             (X2 2)
                             (X1 1)
                             (T 0))
                          (COND
                             (T2 2)
                             (T1 1)
                             (T 0))))
         (RETURN (COND
                    ((AND (NULL ALTFLG)
                          (OR (EQ N 0)
                              FASTYPEFLG))
                     ;; If FASTYPEFLG is T, transpositions are not counted as errors.  Otherwise, transpositions are counted if there are other
                     ;; errors, i.e.  if thee are no errors except for transpostions, SKOR returns 0.0
                     N)
                    (T (IPLUS N NTRANS)))))))
```

(**MOVETOP**

```
  (LAMBDA (X L)                                          (* |Used| |by| |spelling| |block| |and| |helpfixblock.|)
    (PROG ((Y L)
           Z)
      LP   (COND
              ((NULL Y)
               (RETURN L))
              ((NEQ (CAR Y)
                    X)
               (SETQ Z Y)
               (SETQ Y (CDR Y))
               (GO LP))
              ((NEQ Y L)                                 (* |Move| |to| |front| |of| |list|)
               (FRPLACD Z (CDR Y))
               (FRPLACD Y (CDR L))
               (FRPLACD L Y)
               (FRPLACA Y (CAR L))
               (FRPLACA L X)))
          (RETURN L))))
```

)

(RPAQ? **USERWORDS**)

(RPAQ? **SPELLINGS1**
```
       '(DEFINEQ ARGLIST MOVD GETD FNTYP BREAK UNBREAK REBREAK TRACE BREAKIN MAKEFILE MAKEFILES LISTFILES FILES?
                 WHEREIS CLEANUP PP PF EDITF EDITV EDITP ADVISE UNADVISE UNSAVEDEF RECOMPILE TCOMPL COMPILE
                 BRECOMPILE BCOMPL MAPCAR MAPC LOAD LOADFROM LOADFNS TIME CLOSEF CLOSEALL OPENP OUTPUT INPUT
                 OUTFILE INFILE LOGOUT PUTPROP REMPROP GETPROP SYSOUT CLISPIFY DWIMIFY EDITCALLERS FREEVARS CALLS)
```
)

(RPAQ? **SPELLINGS2**
```
       '(GETPROP ADD1 AND APPEND ASSOC COND CONS COPY ELT EQ EQUAL ERROR ERSETQ EVAL FASSOC FMEMB FRPLACA
                 FRPLACD FUNCTION GO IDIFFERENCE IGREATERP ILESSP IMINUS IPLUS ITIMES LENGTH LIST LISTP MAPC
                 MAPCAR MAPCONC MEMB MEMBER NCONC NCONC1 NEQ NLISTP NLSETQ NULL NUMBERP OR PRINT PRIN1 PROG PROGN
                 PUTPROP QUOTE READ RETURN RPLACA RPLACD SELECTQ SETA SETQ SPACES SUB1 TERPRI ZEROP IF F/L VALUEOF
                 FOR FETCH REPLACE CREATE GETPROP PUTPROP DIFFERENCE GREATERP LESSP PLUS))
```

(RPAQ? **SPELLINGS3** '(BROKENFNS ADVISEDFNS NOTLISTEDFILES FILELST NOTCOMPILEDFILES PROMPT#FLG CLISPIFYPRETTYFLG
DWIMIFYCOMPFLG FILERDTBL EDITRDTBL SYSPRETTYFLG NOSPELLFLG INITIALS NIL))

(RPAQ? **SPELLSTR1** "{spellseparator}")

(RPAQ? **SPELLSTR2** "{spellignore}")

(RPAQ? **FIXSPELLREL** 70)

```
(RPAQ? FIXSPELLDEFAULT '\y)

(RPAQ? SKORLST1 '(NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL))

(RPAQ? SKORLST2 '(NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL))

(RPAQ? DWIMKEYLST NIL)

(RPAQ? FIXSPELLKEYLST NIL)

(RPAQ? FASTYPEFLG )

(RPAQ? RUNONFLG NIL)

(RPAQ? \#USERWORDS 20)

(RPAQ? \#SPELLINGS1 20)

(RPAQ? \#SPELLINGS2 20)

(RPAQ? \#SPELLINGS3 20)

(RPAQ? DWIMWAIT 10)

(RPAQ? RESPELLS )

(RPAQ? FIXSPELL.UPPERCASE.QUIET NIL)

(SETSPELLCASE)

(NCONC1 SPELLINGS1 SPELLSTR1)

(NCONC1 SPELLINGS2 SPELLSTR1)

(ATTACH SPELLSTR1 SPELLINGS3)

(DECLARE\: DONTCOPY

(DECLARE\: EVAL@COMPILE

(PUTPROPS SPELLEQ DMACRO (OPENLAMBDA (X Y)
                            (OR (EQ X Y)
                                (AND (< X 255)
                                     (< Y 255)
                                     (EQ (GETCASEARRAY SPELLCASEARRAY X)
                                         (GETCASEARRAY SPELLCASEARRAY Y))))))
)
)

(DECLARE\: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY

(BLOCK\: FIXSPELLBLOCK MISSPELLED? FIXSPELL CHOOZ CHOOZ1 SKOR SKOR0 MOVETOP (ENTRIES MISSPELLED? FIXSPELL CHOOZ
                                                                              SKOR0 SKOR MOVETOP)
       (LOCALFREEVARS NCXWORD NCTWORD TAIL ALTFLG))

(BLOCK\: FIXSPELL1 FIXSPELL1 FIXSPELL2)
)

(DECLARE\: EVAL@COMPILE DONTCOPY

(AND (OR (GETPROP 'NOSPELLFLG 'GLOBALVAR)
         (FMEMB 'NOSPELLFLG GLOBALVARS))
     (HELP "NOSPELLFLG shouldn't be a global variable!" " How did it get that way?"))
)

(DECLARE\: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \#SPELLINGS2 \#SPELLINGS3 \#USERWORDS APPROVEFLG CLISPCHARS CLISPFLG COMMENTFLG DWIMFLG DWIMKEYLST
       DWIMWAIT EDITQUIETFLG FASTYPEFLG LASTWORD REREADFLG RESPELLS RUNONFLG RUNONSTATS SKORLST1 SKORLST2
       SPELLINGS1 SPELLINGS2 SPELLINGS3 SPELLSTATS SPELLSTATS1 SPELLSTR1 SPELLSTR2 USERWORDS VETOSTATS)
)

(PUTPROPS SPELL COPYRIGHT ("Venue & Xerox Corporation" 1984 1985 1986 1990))
```

FUNCTION INDEX

VARIABLE INDEX

MACRO INDEX