File created: 16-May-90 12:28:58 **{DSK}<usr>local>lde>lispcore>sources>CLISPIFY.;2**

changes to: (VARS CLISPIFYCOMS)

previous date: 19-Jun-86 14:57:01 {DSK}<usr>local>lde>lispcore>sources>CLISPIFY.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ CLISPIFYCOMS
       [(FNS CLISPIFYFNS CLISPIFY CLISPIFY1 CLISPIFY2 CLISPIFY2A CLISPIFY2B CLISPIFY2C CLISPIFY2D CLISP3 CLISP3A
             CLISP3B CLISPACKUP CLISP3C CLISP4 CLISPCOND CLISPCOND1 CLISPAND CLISPAND1 CLISPIFYNOT
             CLISPIFYMATCHUP CLREMPARS CLISPIFYCROPS0 CLISPIFYCROPS CLISPIFYCROPS1 CLISPIFYRPLAC CLISPIFYMAPS
             CLMAPS1 CLMAPS2 CLSTOPSCAN? CLISPIFYLOOKUP LOWERCASE SHRIEKIFY SHRKFY SHRKFY2 WHILEDOUNTIL WHILEDO1
             CLDISABLE)
        (INITVARS (FUNNYATOMLST)
               (CLREMPARSFLG)
               (CL%:FLG T)
               (CLISPIFYPACKFLG)
               (CLISPIFYENGLSHFLG)
               (CLISPIFYUSERFN))
        (VARS CAR/CDRSTRING)
        (USERMACROS CL)
        (PROP CLISPFORM ADD1 SUB1 NEQ)
        (PROP CLISPBRACKET CONS LIST APPEND NCONC NCONC1 /NCONC /NCONC1)
        (PROP CLISPTYPE ~EQUAL ~MEMBER ~MEMB)
        (PROP CLMAPS MAPC MAP MAPCAR MAPLIST MAPCONC MAPCON SUBSET)
        (BLOCKS (CLISPIFYBLOCK CLISPIFYFNS CLISPIFY CLISPIFY1 CLISPIFY2 CLISPIFY2A CLISPIFY2B CLISPIFY2C
                       CLISPIFY2D CLISP3 CLISP3A CLISP3B CLISPACKUP CLISP3C CLISP4 CLISPCOND CLISPCOND1 CLISPAND
                       CLISPAND1 CLISPIFYNOT CLISPIFYMATCHUP CLREMPARS CLISPIFYCROPS0 CLISPIFYCROPS
                       CLISPIFYCROPS1 CLISPIFYRPLAC CLISPIFYMAPS CLMAPS1 CLMAPS2 SHRIEKIFY SHRKFY SHRKFY2
                       CLISPIFYLOOKUP CLSTOPSCAN? WHILEDOUNTIL WHILEDO1 (ENTRIES CLISPIFYFNS CLISPIFY CLISPACKUP
                                                                              CLISPIFYMATCHUP CLISPIFY2A
                                                                              CLISP3A)
                       (SPECVARS EXPR VARS DWIMIFYFLG DWIMIFYING DWIMIFY0CHANGE)
                       (LOCALFREEVARS DECLST CLTYP0 OPR0 LST SEG TAIL FORM PARENT SUBPARENT NOVALFLG NEGFLG
                              RESULTP SAFEFLAG VARS CLISPISTATE TYPE-IN? SIDES CLISPIFYFN)
                       (GLOBALVARS CAR/CDRSTRING CL%:FLG CLISPARRAY CLISPCHARRAY CLISPCHARS CLISPFLG
                              CLISPIFYENGLSHFLG CLISPIFYPACKFLG CLISPIFYSTATS CLISPIFYUSERFN CLISPISNOISEWORDS
                              CLISPISVERBS CLISPTRANFLG CLREMPARSFLG COMMENTFLG DWIMFLG FILELST FUNNYATOMLST
                              GLOBALVARS LCASEFLG)
                       (RETFNS CLISPIFY2B)
                       (NOLINKFNS CLISPIFYUSERFN))
              (NIL LOWERCASE (GLOBALVARS CHCONLST LCASEFLG))
              (NIL CLDISABLE (GLOBALVARS CLISPCHARS CLISPCHARRAY NOFIXFNSLST0 NOFIXVARSLST0))
              (NIL (GLOBALVARS CLISPISNOISEWORDS CLISPISVERBS CLISPISWORDSPLST)))
        (P (LOWERCASE T))
        (DECLARE%: DOEVAL@COMPILE DONTCOPY (RECORDS MATCHUP))
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA CLISPIFYFNS)
                                                                              (NLAML)
                                                                              (LAMA])

(DEFINEQ

(CLISPIFYFNS
  [NLAMBDA FNS                                                    (* wt%: 30-JUN-77 22 16)
    (PROG ((CLK (CLOCK 0))
           TEM)
          (RETURN (MAPCONC [COND
                           ((CDR FNS)
                            FNS)
                           ((LISTP (CAR FNS))
                            (STKEVAL 'CLISPIFYFNS (CAR FNS)
                                   NIL
                                   'INTERNAL))
                           (T                                     (* If (CAR FNS) is name of a file, do clipifyfns on its functions.)
                              (OR (LISTP (EVALV (CAR FNS)
                                                'CLISPIFYFNS))
                                  (AND (GETPROP (OR (AND DWIMFLG (MISSPELLED? (CAR FNS)
                                                                              70 FILELST NIL FNS))
                                                    (CAR FNS))
                                                'FILE)
                                       (FILEFNSLST (CAR FNS)))
                                  (STKEVAL 'CLISPIFYFNS (CAR FNS)
                                           'INTERNAL]
                           (FUNCTION (LAMBDA (X)
                                       (COND
                                          ((IGREATERP (IDIFFERENCE (SETQ TEM (CLOCK 0))
                                                                   CLK)
                                                      30000)
```

```
                                        (SETQ CLK TEM)
                                        (PRIN2 X T T)
                                        (PRIN1 '", " T)))
                                    (ERSETQ (CLISPIFY X])
```

(CLISPIFY

```
  [LAMBDA (X EDITCHAIN)                                          (* lmm "27-FEB-83 10:53")
```

(* CLISPIFY the expression X. EDITCHAIN if supplied is the chain of parents of this expression;
used for gathering the variables bound and the top level context)

```
    (PROG (TEM CLISPIFYFN OPR0 CLTYP0 BROADSCOPE DECLST EXPR VARS PARENT SUBPARENT FORM SEG TAIL LST CLISPISTATE
               )
          [COND
             [(OR (LISTP X)
                  EDITCHAIN)
              (COND
                 ((NULL EDITCHAIN)
                  (SETQ EXPR X))
                 (T (SETQ PARENT (CAR EDITCHAIN))
                    (AND (TAILP (SETQ TEM (EVQ LASTAIL))
                                (CAR EDITCHAIN))
                         (SETQ TAIL TEM))
                    [COND
                       ((LISTP (SETQ CLISPIFYFN (EVQ ATM)))      (* New editor conventions.)
                        (SETQ CLISPIFYFN (CAR CLISPIFYFN]        (* ATM is bound in EDITE)
                    (SETQ VARS (VARSBOUNDINEDITCHAIN EDITCHAIN)) (* VARSBOUNDINEDITCHAIN climbs EDITCHAIN and gathers
                    up the VARS)
                    (SETQ EXPR (CAR (LAST EDITCHAIN]
             (T (SETQ TEM (EXPRCHECK X))
                (SETQ CLISPIFYFN (CAR TEM))
                (SETQ EXPR (SETQ FORM (CDR TEM]
          (AND (NULL CLISPIFYFN)
               (SETQQ CLISPIFYFN TYPE-IN))
          (SETQ DECLST (GETLOCALDEC EXPR CLISPIFYFN))
          [COND
             ((NULL FORM)                                        (* Corresponds to first clause in first COND.)
              (RETURN (COND
                         ((NULL EDITCHAIN)                     (* E.G. User just types in CLISPIFY some expression.)
                          (CLISPIFY2 X))
                         ((TAILP X PARENT)
                          (SETQ FORM PARENT)
                          (SETQ TEM (CLISPIFY1 X))
                          (CONS (CAR TEM)
                                (CDR TEM)))
                         ([OR (EQ (CAAR EDITCHAIN)
                                  'COND)
                              (AND (EQ (CAAR EDITCHAIN)
                                       'SELECTQ)
                                   (NEQ X (CADAR EDITCHAIN))
                                   (CDR (FMEMB X PARENT]
                          (SETQ FORM (CAR X))
```

(* The LIST is because while this expression should be CLISPIFIED as a tail
(it being a COND clause)%, it is an element in the structure and the CL macro is expecting a list returned.)

```
                          (LIST (CLISPIFY1 X)))
                         (T                                    (* Commands expect a list returned)
                            (CLISPIFY2A X]
          (SETQ TEM (CLISPIFY2 EXPR))
          (RETURN (COND
                     ((NULL CLISPIFYFN)
                      TEM)
                     (T (COND
                           ((NULL (GETD CLISPIFYFN))
                            (DWIMUNSAVEDEF CLISPIFYFN T)))
                        (/PUTD CLISPIFYFN TEM)
                        (AND FILEPKGFLG (MARKASCHANGED CLISPIFYFN 'FNS))
                        CLISPIFYFN])
```

(CLISPIFY1

```
  [LAMBDA (TAIL OPR0 CLTYP0 BROADSCOPE NOVALFLG SUBPARENT)       (* lmm "27-Jan-85 00:27")
```

(* Processes tails. When OPR0 is not NIL, called from CLISP3, and inserts OPR0 between each call to CLISPIFY2.
If BROADSCOPE is T, OPR0 is an operator with higher precedence than user functions, e.g.
EQ, LS, AND, etc. In this case, the arguments need not be parenthesized, e.g.
(AND (FOO X) Y) -> (FOO X AND Y))

```
    (PROG (SEG TEM LST (TAIL0 TAIL)
               (PARENT FORM)
               PREVEXP)
          [COND
             ((NULL SUBPARENT)
```

(* PARENT and SUBPARENT are used in CLISP4 for checking for partial CLISPIFICATION and resulting calls to DWIMIFY.
e.g. when CLISPIFYING and expression like (FOO X* (IPLUS X Y)) it is necessary to backup and DWIMIFY, but for

(SELECTQ X (X* (IPLUS X Y)) NIL) it is not. PARENT and SUBPARENT are rebound in CLISPIFY1
(rather than CLISPIFY2)%. Therefore they are alsoebound by functions that recurse by calling CLISPIFY2 directly, i.e.
CLISPIFYRPLAC, CLISPIFYCROPS, and CLISPIFYCOND.)

```
        (SETQ SUBPARENT TAIL))
       ((NEQ SUBPARENT TAIL)
        (SETQ PREVEXP (CAR (NLEFT SUBPARENT 1 TAIL]
[COND
   ((EQ NOVALFLG 'NOTLAST)                          (* Says all forms are nor being used for value, EXCEPT for last
    one.)
    (SETQ NOVALFLG (LAST TAIL]
LP [COND
   [(NULL TAIL)
    (RETURN (COND
                ((NULL LST)
                 TAIL0)
                (TAIL0 (NCONC LST TAIL0))
                (T LST]
   [(NLISTP TAIL)
    (RETURN (COND
                (TAIL0 (NCONC LST TAIL0))
                (T (FRPLACD (FLAST LST)
                            TAIL)
                   LST]
   ((AND OPR0 (NEQ OPR0 T))
    (COND
       (LST (NCONC1 LST OPR0))
       ((GETPROP OPR0 'UNARYOP)
        (SETQ LST (LIST OPR0]
[SETQ TEM (COND
             ((AND (LITATOM PREVEXP)
                   (EQ (NTHCHARCODE PREVEXP -1)
                       (CHARCODE %')))
              (CAR TAIL))
             (T (CLISPIFY2 (CAR TAIL)
                       (COND
                          ((EQ NOVALFLG T))
                          (NOVALFLG (NEQ TAIL NOVALFLG]
(SETQ PREVEXP (CAR TAIL))
[COND
   ((OR SEG (NEQ TEM (CAR TAIL))
        OPR0
        (EQ CLTYP0 T))
```

(* The idea in CLISPIFYing is to do as few CONSes as necessary, i.e.
only construct new structure where needed. TAIL0 keeps track of the last point in TAIL for which a corresponding element
has been added to L. Here we know that a new element will hve to be added to L, and so any intermediate elements that
were not added because the clispified result was the same as the original structure, will now have to be added, e.g.
consider (LIST A B (SETQ X Y) C D E)%. The C D and E tail need not be copied, however until we reach
(SETQ X Y) we do not know that the A and B will have to be copied, i.e in
(LIST A B C) no conses are performed. Note that CLTYP0=T is effectively a COPYFLG.
This is used in particular when CLISPIFYING a COND, i.e. OPR0 is NIL, but CLTYP0 isT.)

```
    [AND (NEQ TAIL TAIL0)
         (SETQ LST (NCONC LST (LDIFF TAIL0 TAIL]
    [SETQ LST (COND
                 (SEG
```

(* Supposedly, whenever SEG is set to T, the entire form has already been copied.
An EQ check is not sufficient, as expressions produced by CLIPIFYing may not be EQ to original expressions, but still have
common tails.)

```
                      (NCONC LST TEM))
                 (T (NCONC1 LST TEM]
    (SETQ TAIL0 (CDR TAIL]
(SETQ TAIL (CDR TAIL))
(SETQ SEG NIL)
(GO LP])
```

(CLISPIFY2

```
[LAMBDA (FORM NOVALFLG NEGFLG)                      (* lmm " 5-SEP-83 13:26")
                                                    (* CLISPIFIES a form.)
                                                    (* NOVALFLG is T if FORM is not being used for value.)
  (AND FORM (PROG NIL
                (COND
                   ((AND (LITATOM FORM)
                         OPR0 CLTYP0 (FMEMB FORM CLISPCHARS))
```

(* this check was originally installed because of users who had variables with same name as clisp oprators, and didnt want
them to gt packed.)

```
                    [PROG (POS)
                          (COND
                             ((SETQ POS (STKPOS 'CLISPIFY2B))
                              (RETEVAL POS 'FORM T]          (* CL call on a tail. screw it.)
                    ))
```

```
        LP  (RETURN (OR (CLISPIFY2B FORM)
                        (GO LP)))
```

(* CLISPIFY2b returns NIL (via a RETFROM in CLISP3) when it was necessary to DWWIMIFY the prent expression and staat over, e.g. (FOO X* (IPLUS Y Z)) note that (FOO X * (IPLUS X Y)) wont be touched at all because the * causes an abort)

```
        ])
```

(CLISPIFY2A

```
  [LAMBDA (FORM FLG)
```

(* Whereas the value of CLISPIFY2 is either an item or a segment, depending on SEG, the value of CLISPIFY2A is always a segment, so thatthe caaling function need not check SEG, it can just NCONC or APPEND.)

```
    (PROG (TEM SEG)
          (SETQ TEM (CLISPIFY2 FORM))
          (RETURN (COND
                     ([AND SEG (OR (NULL FLG)
                                   (NULL (CDR TEM]
```

(* If FLG is T, the expression is to be parenthesized, even if SEG is T, unless it is just atomic.)

```
                      TEM)
                     (T (LIST TEM])
```

(CLISPIFY2B

```
  [LAMBDA (FORM)                                        (* lmm "27-Jan-85 00:29")
```

(* Does the work of CLISIPIFY2. This function is separate from CLISPIFY2 so that CLISPIFYNOT can CLISPIFY the inner form, and then check to ee if NEGLFG has been set to NIL. It is also used when a for a %'recursive' call on the same or equivalent form, again so that NEGFLG is not rebound, e.g. (ADD1 --) is the same as (IPLUS -- 1)%, and is implemented by caaling CLISPIFY2b again, itead of CLISPIFY2.)

```
    (PROG (TEM1 TEM2 TEM3)
          [COND
             ((NLISTP FORM)
              (COND
                 ([AND (LISTP CLTYP0)
                       (SETQ TEM1 (GETPROP FORM 'CLISPWORD))
                       (EQ (CAR TEM1)
                           (CAR (GETPROP (CAR CLTYP0)
                                         'CLISPWORD]
                  [AND LCASEFLG (SETQ TEM2 (COND
                                              ((NLISTP (CDR TEM1))
                                               (CDR TEM1))
                                              (T
```

(* The CLISPWORD property can be of the form (FIND find FOR) when FIND is a synonym for FOR.)

```
                                                 (CADR TEM1]      (* Converts FOR words and IF words to loercase.)
                  (SETQ CLTYP0 TAIL)))
              (RETURN (OR TEM2 FORM)))
             ((LISTP (CAR FORM))
              (RETURN (CLISPIFY1 FORM]
          [RETURN (SELECTQ (CAR FORM)
                       (FUNCTION (CLISPIFY2C FORM))
                       (CAR (CLISPIFYCROPS0 '(%:1)))
                       (CDR (CLISPIFYCROPS0 '(|::1|)))
                       ((LAST FLAST)
                            (AND (NEQ (CAR FORM)
                                      (CLISPIFYLOOKUP (CAR FORM)
                                                      (CADR FORM)
                                                      (CADDR FORM)))
                                 (GO A))
                            (CLISPIFYCROPS0 (LIST -1)))
                       (NLEFT (COND
                                 [(AND (NUMBERP (SETQ TEM1 (CADDR FORM)))
                                       (NULL (CDDDR FORM)))
                                  (CLISPIFYCROPS0 (LIST (MINUS TEM1]
                                 (T (GO A))))
                       ((NTH FNTH)                              (* (NTH X 10) clispifies to |X::9|)
                            (COND
                               [[AND CL%:FLG (NUMBERP (CADDR FORM))
                                     (EQ (CAR FORM)
                                         (CLISPIFYLOOKUP (CAR FORM)
                                                         (CADR FORM]
                                (CLISPIFYCROPS0 (LIST (SUB1 (CADDR FORM]
                               (T (GO A))))
                       ((RPLACA FRPLACA /RPLACA)
                            (CLISPIFYRPLAC FORM '%:1 T))
                       ((RPLACD FRPLACD /RPLACD)
                            (CLISPIFYRPLAC FORM '|::1| T))
                       ((CLISP%: GO DECLARE)
                            FORM)
```

```
                    (* (COND
                          ((EQ (CADR FORM)
                               'DECLARATIONS%:)
                           (CONS 'CLISP%: (CDDR FORM)))
                          (T FORM)))
                    (COND (COND
                             ((NULL (GETP 'IF 'CLISPWORD))
                              (GO A))
                             [(CDR FORM)
                              (FRPLACA (PROG ((L FORM)
                                              VAL)            (* An open MAPCONC.)
                                             LP   (COND
                                                     ((NULL (SETQ L (CDR L)))
                                                      (RETURN VAL)))
                                                  (SETQ VAL (NCONC VAL (CLISPCOND (CAR L)
                                                                                  (CDR L)
                                                                                  VAL)))
                                                  (GO LP))
                                       (COND
                                          (LCASEFLG 'if)
                                          (T 'IF]
                              (T FORM)))
                    ((AND OR)
                         (COND
                            ((NULL (GETP 'IF 'CLISPWORD))
                             (GO A))
                            (NOVALFLG                         (* Treat AND as COND.)
                                   (CLISP4 PARENT SUBPARENT)
                                   (SELECTQ (CAR FORM)
                                        (AND [FRPLACA (CLISPAND FORM)
                                                      (COND
                                                         (LCASEFLG 'if)
                                                         (T 'IF])
                                        (OR (COND
                                               [(NULL (CDDDR FORM))
                                                (FRPLACA (CLISPCOND (CONS (LIST 'NOT (CADR FORM))
                                                                          (CDDR FORM)))
                                                         (COND
                                                            (LCASEFLG 'if)
                                                            (T 'IF]
                                                (T (GO A))))
                                            (SHOULDNT)))
                            (T (GO A))))
                    (SELECTQ [PROG (OPR0 CLTYP0 PARENT SUBPARENT)
                                   (CLISP4 FORM)
                                   (SETQ PARENT FORM)
                                   (SETQ SUBPARENT (CDR FORM))
                                   (SETQ TEM1 (CLISPIFY2A (CADR FORM)
                                                          T))
                                   [SETQ TEM2 (AND (CDDDR FORM)
                                                   (MAPCAR (CDDR FORM)
                                                           [FUNCTION (LAMBDA (FORM)
                                                                             (COND
                                                                                [(LISTP FORM)
                                                                                 (CONS (CAR FORM)
                                                                                       (CLISPIFY1 (CDR FORM)
                                                                                                  NIL NIL NIL
                                                                                                  (OR NOVALFLG
                                                                                                      'NOTLAST]
                                                                                (T FORM]
                                                           (FUNCTION (LAMBDA (X)
                                                                             (AND (CDDR X)
                                                                                  (CDR X]
                                   (RETURN (CONS 'SELECTQ (NCONC TEM1 TEM2 (CLISPIFY2A (CAR (LAST FORM))
                                                                                       T]))
                    (PROGN (SETQ TEM1 (CLISPIFY1 (CDR FORM)
                                                 NIL NIL NIL 'NOTLAST))
```

(* novalflg used to be (OR NOVALFLG (QUOTE NOTLAST)) however, this caused a bug in the case where one had FOO_ (PROGN --) at the top level, because the FFO_ never got translated because the PROGN didnt require it.
rather than fix this, obseeve that if in fact PROGN were in noval context, then the user would not need a progn at all, so lets just assume that a progn is always in value context, and specify NOTLAST in the call to clispify1)

```
                           (CLISPIFY2C FORM TEM1))
                    (NULL [COND
                             ((AND (LISTP (CADR FORM))
                                   (GETPROP (SETQ TEM1 (CAADR FORM))
                                            'CLISPTYPE))          (* e.g. (NULL (NUMBERP X)) is treated as
                                                                  (NOT (NUMBERP X)))
                              (CLISPIFYNOT (CADR FORM)))
                             (T
```

(* reason for not simply resetting form and jumping to top has to do with the way clispify handle partially clispifyed expressions namely, by dwimifying the expression, retfroming NIL from cal to clispify2b, and having clispify2 then try again, relying on the fact that dwimify has physically changed form. If we just reset form here, and there was any clisp in the original form, an infinite loop wouldoccur since clipify2 would keep retrying with original form.)

```
                    (CLISPIFY2 (LIST 'EQ (CADR FORM)
                                     NIL)
                               NOVALFLG
                               (PROG1 NEGFLG (SETQ NEGFLG])
               (NOT (CLISPIFYNOT (CADR FORM)))
               (SETQ (SETQ VARS (CONS (CADR FORM)
                                      VARS))          (* In case any dwimifying occurs as a result of partial
                                                         clispification.)
                     [AND (CDDDR FORM)
                          (SETQ FORM (LIST (CAR FORM)
                                           (CADR FORM)
                                           (CONS 'PROG1 (CDDR FORM]
                     (GO A))
               (SETQQ (COND
                          ((AND CLISPFLG (GETPROP '_ 'CLISPTYPE))
                           (CLISPIFY2 [LIST 'SETQ (CADR FORM)
                                            (COND
                                               ((OR (NULL (SETQ TEM1 (CADDR FORM)))
                                                    (NUMBERP TEM1))
                                                TEM1)
                                               (T (LIST 'QUOTE TEM1]
                                      NOVALFLG NEGFLG))
                          (T (GO B))))
               ((match MATCH)
                   [PROG ((OPR0 (AND [NULL (CDR (SETQ TEM2 (CDDDR FORM]
                                     OPR0)))
```

(* OPR0 rebound to NIL if -> or => used in match expression, because in this case, want CLISPIFYCROPS to %'wrap it up'.)

```
                         (SETQ TEM1 (CLISPIFYCROPS (CADR FORM)
                                                   (LIST (CAR TEM2))
                                                   'match]
                     [COND
                        ((NULL (SETQ TEM2 (CDR TEM2)))
                         TEM1)
                        (T (NCONC TEM1 (CONS (CAR TEM2)
                                             (CLISPIFY1 (CDR TEM2])
               ((fetch FETCH)
                   (COND
                      ([AND (EQLENGTH FORM 4)
                            (FMEMB (CADDR FORM)
                                   '(of OF]
                       (CLISP4 FORM)
                       (CLISPIFYCROPS (CADDDR FORM)
                              (LIST (CADR FORM))
                              'fetch))
                      (T (GO A))))
               ((replace REPLACE)
                   (COND
                      ([AND (EQLENGTH FORM 6)
                            (OR (EQ (SETQ TEM1 (CADDR FORM))
                                    'OF)
                                (EQ TEM1 'of))
                            (OR (EQ (SETQ TEM1 (CAR (CDDDDR FORM)))
                                    'WITH)
                                (EQ TEM1 'with]
                       (CLISPIFYRPLAC FORM 'replace T))
                      (T (GO A))))
               (ASSEMBLE FORM)
               (COND
                  ((EQ (CAR FORM)
                       CLISPTRANFLG)
                   (SETQ FORM (CDDR FORM))
                   (CLISPIFY1 FORM NIL FORM))
                  ((FMEMB (CAR FORM)
                          CLISPCHARS)
                   (RETURN FORM))
                  [(AND (OR (EQMEMB 'BINDS (GETPROP (CAR FORM)
                                                    'INFO))
                            (FMEMB (CAR FORM)
                                   LAMBDASPLST))
                        (NOT (CLISPIFY2D FORM)))          (* lambda, nlambda, dlambda, prog, resetvars, etc.)
                   (CONS (COND
                            [(AND [LISTP (SETQ TEM2 (GETP (CAR FORM)
                                                          'CLISPWORD]
                                  LCASEFLG)
                             (COND
                                ((NLISTP (CDR TEM2))
                                 (CDR TEM2))
                                (T (CADR TEM2]
                            (T (CAR FORM)))
                         (CONS [COND
                                  ((NULL (CADR FORM))
                                   NIL)
                                  ((NLISTP (CADR FORM))
```

(* This is a quick and dirty attempt to collect vars in cse have to call DWIMIFY1B.
VARS are not rebound each expression, so open lambda variables will justbuild up.

if this turns out to be a problem, will have to rebind vars each time we call clispify)

```
                                        (SETQ VARS (CONS (CADR FORM)
                                                         VARS))
                                        (CADR FORM))
                                       (T (MAPCAR (CADR FORM)
                                                  (FUNCTION (LAMBDA (X)
                                                              (COND
                                                                 ((NLISTP X)
                                                                  (SETQ VARS (CONS X VARS))
                                                                  X)
                                                                 (T (SETQ VARS (CONS (CAR X)
                                                                                     VARS))
                                                                    (CONS (CAR X)
                                                                          (CLISPIFY1 (CDR X]
                                    (CLISPIFY1 (WHILEDOUNTIL (CDDR FORM))
                                           NIL
                                           (AND (GETP (CAR FORM)
                                                      'CLISPWORD)
                                                FORM)
                                           NIL
                                           (COND
                                              ((MEMB 'LABELS TEM1)
                                                                  (* e.g. prog no member used for value)
                                               T)
                                              ((MEMB 'PROGN TEM1)
                                                                  (* e.g. lambdas, nlambdas. only last element is used for value)
                                               (OR NOVALFLG 'NOTLAST))
                                              (T                  (* e.g. for %, bind)
                                                 NIL]
                    (T (GO A]
A   [COND
       [(AND (SETQ TEM1 (GETPROP (CAR FORM)
                                 'CROPS))
             (NEQ (CAR FORM)
                  'GETPROPLIST))
        (RETURN (CLISPIFYCROPS0 (SUBPAIR '(A D)
                                         '(%:1 |::1|)
                                         TEM1]
       [[AND (SETQ TEM1 (GETPROP (CAR FORM)
                                 'CLISPCLASS))
             (SETQ TEM3 (GETPROP TEM1 'CLISPTYPE]          (* E.G. (CAR FORM) is FPLUS, TEM1 is +.)
        (COND
           ([EQ (CAR FORM)
                (CLISPIFYLOOKUP (CAR FORM)
                       (CADR FORM)
                       (CADDR FORM)
                       TEM1
                       (GETPROP TEM1 'CLISPCLASSDEF]
            (RETURN (CLISP3 (OR (GETPROP (GETPROP TEM1 'LISPFN)
                                         'CLISPINFIX)
                                TEM1)
                            FORM TEM3))
```

(* TEM1 is now for example LT. Reason for not siply passing LT is this permits user to put lower case lt on property lst of ILESSP under clispifnix property.)

```
           ]
       ((AND (SETQ TEM1 (GETPROP (CAR FORM)
                                 'CLISPINFIX))
             (CDDR FORM)
             (OR NEGFLG OPR0 CLTYP0 (FMEMB TEM1 CLISPCHARS)))
```

(* E.g. IF (CAR FORM) is EXPT, TEM1 would be ^. The CLTYP0 is because only want to convert to infix if under another operator, e.g. (LIST (AND X Y)) is clearer than (LIST (X AND Y)))

```
        (RETURN (CLISP3 TEM1 FORM (GETPROP TEM1 'CLISPTYPE]
    [RETURN (COND
               ((SETQ TEM1 (GETPROP (CAR FORM)
                                    'CLISPFORM))          (* E.G. NEQ, ADD1, and SUB1.)
```

(* code used to say (GO TOP)%. Then was changed to call Clispify2.
Callin clipify2 has a bad effect when negflg is t as it introduces an extra binding of negflg.)

```
                 (CLISPIFY2B (LSUBST (CDR FORM)
                                     '* TEM1)))
                [[AND (SETQ TEM1 (GETPROP (CAR FORM)
                                          'CLISPBRACKET))
                      (SETQ TEM2 (GETPROP TEM1 'CLISPBRACKET]
                 (COND
                    [(SETQ TEM3 (LISTGET1 TEM2 'CLISPIFY))  (* built in userfn)
                     (COND
                        ((EQ TEM3 'SHRIEKIFY)
                         (COND
                            ([OR (NULL CLISPFLG)
                                 (NULL (GETPROP TEM1 'CLISPTYPE))
                                 NOVALFLG
```

```
                                    (NULL (SETQ TEM1 (PROG ((PARENT FORM))
                                                           (RETURN (SHRIEKIFY FORM]
                                     (GO B))
                                    ((EQ (CAR TEM1)
                                         '<)
                                     (CLISP3 '< TEM1 'BRACKET T))
                                    (T                                    (* E.G. didnt convert because of declarations.)
                                       TEM1)))
                                (T (SETQ TEM1 (APPLY* TEM3 FORM]
                               ((GETP TEM1 'UNARYOP)
                                (SETQ TEM3 (CLISPIFY1 (CDR FORM)
                                                      (OR (SETQ TEM3 (LISTGET1 TEM2 'SEPARATOR))
                                                          T)
                                                      'BRACKET))
                                (CLISP3 TEM1 [CONS (CAR TEM2)
                                                   (APPEND TEM3 (LIST (CADR TEM2]
                                           'BRACKET T))
                               (T (SETQ TEM3 (CLISPIFY1 (CDDR FORM)
                                                        (OR (SETQ TEM3 (LISTGET1 TEM2 'SEPARATOR))
                                                            T)
                                                        'BRACKET))
                                  (CLISP3 TEM1 [CONS (CLISPIFY2 (CADR FORM))
                                                     (CONS (CAR TEM2)
                                                           (APPEND TEM3 (LIST (CADR TEM2]
                                           'BRACKET T]
                          [[LISTP (SETQ TEM1 (GETPROP (CAR FORM)
                                                      'SETFN]
```

(* The third aagument to CLISPIFYRPLAC indicates this is a %: transformation.
It is also true if there is an ACCESSFN property. E.g. FOO has ACCESSFN GETFOO SETFN SETFOO and SETFOO has SETFN (FOO))

```
                (CLISPIFYRPLAC FORM (CAR TEM1)
                       (GETPROP (CAR TEM1)
                                'ACCESSFN]
                ((AND TEM1 (EQ (SETQ TEM2 (GETPROP (CAR FORM)
                                                   'ACCESSFN))
                               (CAR FORM)))                       (* Occurs when FOO is its own accessfn, e.g.
                                                                  FOO has ACCESSFN FOO SETFN SETFOO.)
                 (CLISPIFYCROPS0 (LIST TEM2)))
                ([LISTP (SETQ TEM1 (GETPROP (CAR FORM)
                                            'ACCESSFN]
                 (CLISPIFYCROPS0 TEM1))
                [(AND (SETQ TEM1 (GETPROP (CAR FORM)
                                          'CLMAPS))
                      (CLISPIFYMAPS (CAR TEM1)
                             (CDR TEM1]
                ((AND (LITATOM (CAR FORM))
                      (NULL (FGETD (CAR FORM)))
                      (GETPROP (CAR FORM)
                             'CLISPWORD)
                      (NOT (CLISPIFY2D FORM)))
                 (CLISPIFY1 FORM NIL FORM))
                ((CLISPNOEVAL (CAR FORM)
                        T)                                        (* Dont clispify the tails of nlambdas that dont evaluate their
                                                                  arguments.)
                 FORM)
                ((AND CLISPIFYUSERFN (SETQ TEM1 (CLISPIFYUSERFN FORM)))
                 TEM1)
                ((AND [COND
                         [(LITATOM (CAR FORM))
                          (NULL (FGETD (CAR FORM]
                         ((LISTP (CAR FORM))
                          (NULL (OR (EQ (CAAR FORM)
                                        'LAMBDA)
                                    (EQ (CAAR FORM)
                                        'NLAMBDA]
                      (GETHASH FORM CLISPARRAY))
                 (PUTHASH FORM NIL CLISPARRAY)
                 (CLISPIFY2B FORM))
                ((NULL (CDR FORM))
```

(* NULL checks for No arguments, so must leve as item since otherwise would be convereted by dwimify to a variable, e.g. (EQ (FOO) FORM) cannot become FOO=X. The AND checks for nlambdas.)

```
                 FORM)
                (T (GO B]
B
```

(* On this call subparent is specified as being the fomr itself because OF cases like (x* (IPLUS X Y)))

```
(RETURN (CLISPIFY2C FORM NIL FORM])
```

(CLISPIFY2C

```
  [LAMBDA (FORM X SUBPARENT)                                      (* lmm "27-FEB-83 10:38")
```

(* (CAR FORM) is not to be treated specially. CLISPIFY2C simply calls CLISPIFY1.)

```
    (OR X (SETQ X (CLISPIFY1 (CDR FORM)
                        NIL NIL NIL (COND
                                       ((EQMEMB 'PROGN (GETPROP (CAR FORM)
                                                                'INFO))
                                        'NOTLAST))
                        SUBPARENT)))
    (COND
       ((NEQ X (CDR FORM))
        (CONS (CAR FORM)
              X))
       (T FORM])
```

(CLISPIFY2D

```
  [LAMBDA (FORM)                                                    (* wt%: "23-JUL-78 23:32")
```

(* expressions like (SUM + X) do not translate into iterative statements,
(see wtfix1) so that when clipified, they should not be lowercased.
this function returns T if the second element of an the form, X, would cause the expression not to dwimify as an i.s.)

```
    (PROG (TEM)
          (RETURN (AND (SETQ TEM (CADR FORM))
                       (LITATOM TEM)
                       (OR (GETPROP TEM 'CLISPTYPE)
                           (MEMB (SETQ TEM (NTHCHAR TEM 1))
                                 CLISPCHARS))
                       (NOT (GETPROP TEM 'UNARYOP))
                       [NOT (BOUNDP (SETQ TEM (CADR FORM]
                       (NOT (MEMB TEM VARS))
                       (NOT (MEMB TEM NOFIXVARSLST))
                       (NOT (GETPROP TEM 'GLOBALVAR))
                       (NOT (MEMB TEM GLOBALVARS])
```

(CLISP3

```
  [LAMBDA (OPR X CLTYP FLG)                                         (* lmm " 5-SEP-83 23:53")
    (PROG (L (BROADSCOPE (GETPROP OPR 'BROADSCOPE))
             TEM CLISPISTATE)
          [COND
             ((OR (NULL CLTYP)
                  (NULL CLISPFLG))
```

(* This permits user to disable CLISPIFY transformations and CLISP transformaions simply by remving CLISPYTPE property)

```
              (RETURN (CLISPIFY2C X]
          (SETQ L (CDR X))
          (COND
             (FLG                                                   (* X was alrady CLISPIFIED. Used by CLISPIFYNOT)
                  (SETQ L X)
                  (GO OUT))
             ((EQ OPR '%')
              (SETQ L (LIST OPR (CAR L)))
              (GO OUT))
             ([AND NEGFLG [OR (NULL CLISPIFYENGLSHFLG)
                              (NULL (GETPROP OPR 'CLISPIFYISPROP]
                   (SETQ TEM (GETPROP OPR 'CLISPNEG]
              (SETQ NEGFLG NIL)
              (SETQ FLG T)
```

(* FLG is set so that clisp3 can know that negflg ws turned off in case for some reason it was unable to convert to clispify, e.g. variable was also name of function.)

```
              (SETQ OPR TEM)))
          [AND (NULL (GETPROP OPR 'UNARYOP))
               (NULL (CDR L))
               (COND
                  [(EQ (ARGTYPE (GETP OPR 'LISPFN))
                       2)                                           (* E.G. (IPLUS X))
                   (RETURN (CONS (CAR X)
                                 (CLISPIFY1 (CDR X]
                  (T (SETQ L (LIST (CADR X)
                                   NIL]
          (AND PARENT (CLISP4 PARENT SUBPARENT))
```

(* e.g. |...| x* (iplus y z) need to dwimify the higher expression in order to discover the X* and to know that (iplus y z) must be parenthesized)

```
          (CLISP4 X)                                                (* e.g. (iplus x _ (exp) z))
          (SETQ L (CLISPIFY1 L OPR CLTYP BROADSCOPE))
    OUT   (COND
             ([OR (EQ (CAR L)
                      '-)
                  (AND (NUMBERP (CAR L))
                       (MINUSP (CAR L]                              (* Unary minus must be parenthesized)
              (SETQ SEG NIL))
             [(NULL OPR0)                                           (* Parent form is a regular function)
```

```
            (SETQ SEG (COND
                        (CLTYP0
```

(* Parent form is an IF or FOR. Pathological cases occur when clispifying an already paatially clispified expression, e.g. (IF a then (AND B C) D)%. Here cant remove parentheses, but in (IF (AND A B) THEN C) you can)

```
                               (COND
                                  ((OR (EQ CLTYP0 'COND)
                                       (EQ CLTYP0 'IS))
```

(* Started out with a COND. CLISPCOND is careful about setting CLTYP0 so it is safe to remove parentheses)

```
                                    T)
                                   ((NLISTP CLTYP0)
                                    (SHOULDNT))
                                   ([AND (EQ TAIL (CDR CLTYP0))
                                         [OR (NULL (CDR TAIL))
                                             (EQ (CAR (GETPROP (CADR TAIL)
                                                               'CLISPWORD))
                                                 (CAR (GETPROP (CAR CLTYP0)
                                                               'CLISPWORD]
                                         (NULL (SOME L (FUNCTION (LAMBDA (X)
                                                                   (LISTP (GETPROP X 'CLISPWORD]
```

(* Says there is only one expression there so safe to remove parentehses, e.g. (IF A THEN (AND B C) ELSE D) The reason for the SOME is that cant remve parens if any of the words in L are also operators, e.g. user writes (WHILE (IGREATERP X COUNT) do --) can't remove parens because COUNT is also an operator)

```
                                     T)))
                              (BROADSCOPE
```

(* If BROADSCOPE is T, form must be parentheseized, e.g. (FOO (AND X Y)) must be (FOO (X AND Y)) not (FOO X AND Y))

```
                                    NIL)
                              ((NULL CLISPIFYPACKFLG)
                               (NOT (CLISPNOEVAL (CAR PARENT)
                                         T)))
                              (T T]
                 ((OR (AND (LITATOM (CAR L))
                           (FGETD (CAR L)))
                      (CLISP3B OPR CLTYP))
                  (SETQ SEG NIL))
                 (T (SETQ SEG (COND
                                ((AND BROADSCOPE (EQ CLTYP0 'BRACKET))
                                 NIL)
                                (T T)))                    (* And packing will be done by higher operator)
                    (RETURN L)))
(SETQ TEM (CLISP3A L))
(RETURN (COND
           ((AND (LITATOM (CAR L))
                 (CLISPNOEVAL (CAR L)))
```

(* Kaplan insists on this check. He has variables and nlambda function of the same name. Note if function isnot defined at clispify time, you lose, i.e. (AND FOO X) will go to (FOO AND X))

```
            (SETQ NEGFLG FLG)                          (* SEE COMMENT AT CHECK FOR NEGFLG EARLIER.)
            (SETQ SEG NIL)
            (CLISPIFY2C X))
           (T TEM])
```

(CLISP3A

```
[LAMBDA (L)                                            (* lmm "27-FEB-83 09:14")
```

(* L is a list of operands and operators. CLISP3A packs up the atoms. Its value is always a list. CLISPCHARS is a list of those infix operators which can be packed with their operands. Most of these are single characters, but for example ~= appears on this list.)

```
   (SETQ L (CLISPACKUP L))
   (COND
      ([SELECTQ (CAR PARENT)
            ((SELECTQ SETN)                            (* if packing results in more than one expression, then must put
                                                       parens around it.)
                (CDR L))
            (EQMEMB 'LABELS (GETPROP (CAR PARENT)
                                     'INFO]
       (SETQ SEG NIL)))
   L])
```

(CLISP3B

```
[LAMBDA (OPR CLTYP)                                    (* wt%: " 2-JUL-78 18:07")
```

(* called by clisp3 and clispifycrops. determines if parens are needed around the operator cluster by checking whether higher operator would incorrectly gobble parts of this one OR is true if inner operaton must be parenthesized.
First clause corresponds to case where there is an operand to the left of this one, and the inner operator would stop the scan of the outer one, i.e. the one on the left, e.g. (ITIMES A (IPLUS B C))%, must go to A*
(B+C)%. Second clause corresponds to case where there is an operand to the right of this one, and the outer operator, i.e. the one on the right, would NOT stop the inner, e.g. (ITIMES (IPLUS A B) C)%, mustgo to
(A+B) *C.)

(* The AND LISTP ILESSP expression is to cover the case handled specially in stopscan?, namely that of A*B_Y+C grouping as A* (B_Y+C) because the right precedence of _ is looser than that of *.
This is handled here by making the same comparison. It will result in some extra unnecessary parens in some cases, e.g. (ITIMES A (SETQ B (IPLUS C D))) clispifies to (A* (B_Y+C))%.
However, note that in (IPLUS (ITIMES A (SETQ B Y)) C)%, the parens in
(A* (B_Y) +C) ARE necessary. However, the information relating to this is TWO tails above this operator, so better just to be safe.)

```
(AND OPR0 (NEQ CLTYP0 'BRACKET)
     (NEQ CLTYP 'BRACKET)
     (OR [AND LST (OR (CLSTOPSCAN? CLTYP CLTYP0)
                      (AND (LISTP CLTYP)
                           (ILESSP (CDR CLTYP)
                                   (COND
                                      ((ATOM CLTYP0)
                                       CLTYP0)
                                      (T (CDR CLTYP0]
         (AND (CDR TAIL)
              (NOT (CLSTOPSCAN? CLTYP0 CLTYP])
```

(CLISPACKUP

```
[LAMBDA (L)                                                    (* wt%: " 9-JAN-80 20:47")
  (PROG ((LL L)
         L1 L2 TEM L-1 OPRFLG)
    TOP [COND
           ((NOT (ATOM (CAR LL)))
            (SETQ L-1 NIL)
            (GO PACKUP))
           ((EQ (CAR LL)
                '%')                                           (* %' has to be handled specially)
            [COND
               ((OR (NOT (FMEMB (CAR L2)
                                CLISPCHARS))
                    (EQ (CAR L2)
                        '!))
```

(* If the previous element was NOT an operatr, the %' must start a separate atom, therefore pack up the segment up to the %'.)

```
            (CLISP3C L1 L2)
            (AND (ATOM (CADR LL))
                 (CLISP3C LL (CDR LL)))
```

(* The %' and its argument must also be packed up, even if other opeators follow, because %' is always the last operaar in an atom.)

```
            )
           (T (SETQ LL (CLISP3C L1 (COND
                                      ((ATOM (CADR LL))
                                       (CDR LL))
                                      (T LL]
       (SETQ L1 (SETQ L-1 NIL))
       (GO LP1))
      [(OR (NOT (FMEMB (CAR LL)
                       CLISPCHARS))
           (EQ (CAR LL)
               '!))
       (COND
          ((NOT (LITATOM (CAR LL)))
           (SETQ L-1 LL))
          ((FMEMB 'CLISPTYPE (GETPROPLIST (CAR LL)))
```

(* FMEMB is used instead of GETPROP so that we can tell CLISP3A not to pack up thinks like ~EQUAL without making them be oprators.)

```
            (SETQ L-1 NIL)
            (GO PACKUP))
           ([COND
               (FUNNYATOMLST
```

(* The STRPOSL in the next clause slows CLISPIFY down about 10 per cent, but is necessary to catch funnyatoms.
If the user specifies FUNNYATOMLST, its a little faster.)

```
                    (AND (NEQ FUNNYATOMLST T)
                         (FMEMB (CAR LL)
                                FUNNYATOMLST)))
               (T (STRPOSL CLISPCHARRAY (CAR LL]
```

(* The STRPOSL prevents a %'funny atom' from being packed with another operator, e.g.
(IPLUS *X Y) goes to *X +Y. Setting OPRFLG to NIL will cause us to GO to PACKUP.)

```
        (SETQ L-1 NIL)
        (GO PACKUP))
       (T (SETQ L-1 LL)))
    (COND
       (OPRFLG                                    (* OPRFLG is T if previous element was a CLISPCHAR.
                                                  Therefore, continue scanning this cluster.)
              (GO LP1))
       (T (GO PACKUP]
   ((SETQ TEM (GETPROP (CAR LL)
                       'CLISPBRACKET))
    (COND
       [(AND (EQ (CAR LL)
                 (CAR TEM))
             (NEQ (CAR L2)
                  (CAR TEM)))
        (COND
           ((GETPROP (CAR LL)
                     'UNARYOP)
            (CLISP3C L1 L2)
            (SETQ L1 (SETQ L-1 NIL]
       ((AND (EQ (CAR LL)
                 (CADR TEM))
             (NEQ (CADR LL)
                  (CADR TEM)))
        (SETQ LL (CLISP3C (COND
                          ((OR (EQ (CAR (SETQ TEM (OR L1 L-1)))
                                   '+)
                               (EQ (CAR TEM)
                                   '-))
                           (CDR TEM))
                          (T TEM))
                    LL))
        (SETQ L1 (SETQ L-1 NIL))
        (GO LP1]                                  (* At this point we know that the current element is a
                                                  CLISPCHAR.)
(AND (NULL L1)
     (SETQ L1 (OR L-1 LL)))
```

(* L1 marks the beginning of the sequence of atoms to be packed.
If L-1 is not NIL, the atom before this one is not a CLISP word and is to be included, e.g.
(A + B) Note that we don't want to set L1 until we do see a CLISPCHAR, e.g.
(AND A B (EQ X Y)) beecomes (A AND B AND X=Y)%. This is why we must save the last non-operator on L-1 until this point.)

```
(COND
   ((EQ (CAR LL)
        '+)
    (AND (EQ (CADR LL)
             '-)
         (FRPLACA LL '+-)
         (FRPLACD LL (CDDR LL)))
```

(* This simplifies the code as it allows (IPLUS -- (IMINUS --) --) to be treated the same as IDIFFERENCE.)

```
    )
   ((NEQ (CAR LL)
         '+-)
    (GO A)))
```

(* At this point we know (CAR L) is either a + or a +-. (+- is the symbol for binary miinus.) The next COND checks for some special cases.)

```
[COND
   [(AND L-1 (NUMBERP (SETQ TEM (CADR LL)))
         (MINUSP TEM))
    (AND [COND
            ((EQ (CAR LL)
                 '+-)
```

(* E.g. (IDIFFERENCE X -3) -> X +- -3 so change the +- to + and reverse the sign.)

```
             (FRPLACA LL '+))
            ((GETPROP '- 'CLISPTYPE)
```

(* E.g. (IPLUS X -3) -> X + -3, so change the + to -
and reverse the sign. The GETPROP is because -
may be disabled, and wold not be detected beyond this point. Note that if user disables -, he should also disable +-.)

```
             (FRPLACA LL '-]
         (FRPLACA (CDR LL)
                  (MINUS TEM]
   ((EQ (CAR LL)
        '+-)
    (COND
```

```
                ((EQ (CADR LL)
                     '-)

        (* E.g. (IDIFFERENCE X (IMINUS &)) -> X +- -&, so change the +-
        to + and reverse the sign)

                 (FRPLACA LL '+)
                 (FRPLACD LL (CDDR LL)))
                (T (FRPLACA LL '-]
     A  (SETQ L-1 NIL)
        (SETQ OPRFLG T)
        (GO LP)
     LP1 (SETQ OPRFLG NIL)
     LP  (SETQ L2 LL)                                    (* L2 stays one behind L0.)
        (COND
           ((NULL LL))
           ((SETQ LL (CDR LL))
            (GO TOP))
           (T (GO PACKUP)))
        (RETURN L)
     PACKUP
        (AND L1 (CLISP3C L1 L2))
        (SETQ L1 NIL)
        (GO LP1])
```

(CLISP3C

```
  [LAMBDA (L1 L2)                                      (* lmm "24-DEC-81 13:28")
    (PROG (TEM (L3 (CDR L2)))
          (COND
             ((NULL L1)
              (RETURN L2))
             ((EQ L2 L1)
              (RETURN L1)))
          (FRPLACD L2)
          [COND
             (CLISPIFYPACKFLG [SETQ TEM (RESETVARS (PRXFLG)
                                                   (RETURN (PACK L1]
                    (COND
                       ((AND [COND
                                (SEG (NOT (BOUNDP TEM)))
                                (T (NOT (FNTYP TEM]
                             (NOT (FMEMB TEM FUNNYATOMLST))
                             (NOT (NUMBERP TEM)))
```

(* The FMEMB prevents packing up (ITIMES A B) ino A*B if A*B is a %'funny atom', since in this case it wold not ever be unpacked by DWIM. The check for clipifypackflg is made here rather than in clisp3a before calling clispackup because the smarts for converting +- to -
are in clispackup.)

```
                     (FRPLACA L1 TEM)
                     (FRPLACD L1 L3)
                     (RETURN L1]
          (RETURN (FRPLACD L2 L3])
```

(CLISP4

```
  [LAMBDA (EXP SUBPARENT)                              (* lmm "19-Jun-86 14:26")
```

(* CLISP4 is called when an xpression is abut to be converted into infix notation.
IN this case, both the interior of the expression, and its parent, are examined.
SUBPARENT is used when checking the parent, i.e. exp = parent, to prvent backing up to far, e.g.
(SELECTQ (= (SETQ X 5)) T))

```
    (COND
       ([AND CLISPFLG (NEQ (CAR EXP)
                           'SETQ)
             [NOT (EQMEMB 'LABELS (GETPROP (CAR EXP)
                                           'INFO]
             (NULL (GETHASH EXP CLISPARRAY))
             (SOME (OR SUBPARENT EXP)
                   (FUNCTION (LAMBDA (X $TAIL)
                               (AND (LITATOM (CAR $TAIL))
                                    (NOT (BOUNDP (CAR $TAIL)))
                                    (NOT (FMEMB (CAR $TAIL)
                                                VARS))
                                    [NULL (AND (EQ $TAIL EXP)
                                               (FGETD (CAR $TAIL]
                                    (PROG ((N 1))
                                      LP  (COND
                                             ((NULL (SETQ N (STRPOSL CLISPCHARRAY (CAR $TAIL)
                                                                     N)))
                                              (RETURN NIL))
                                             ((GETPROP (NTHCHAR (CAR $TAIL)
                                                                N)
                                                       'CLISPTYPE)
                                              (RETURN N)))
```

```
                                        (SETQ N (ADD1 N))
                                        (GO LP]
        (PROG (POS FLG)
              [PROG ((DWIMIFYFLG 'CLISPIFY)
                     (NOSPELLFLG T))
                    (SETQ FLG (DWIMIFY0? EXP (OR SUBPARENT EXP)
                                         NIL NIL NIL CLISPIFYFN 'LINEAR]
          LP  (COND
                 [(NULL (SETQ POS (STKPOS 'CLISPIFY2B -1 POS POS)))
                                                        (* Can occur if user calls CL at funny)
                  (RETFROM 'CLISPIFY (APPLY 'CLISPIFY (STKARGS 'CLISPIFY]
                 ((OR (NEQ (STKEVAL POS 'FORM)
                           EXP)
                      (NOT FLG))
                  (RETEVAL POS 'FORM T))
                 (T (RETFROM POS NIL T])
```

(CLISPCOND

```
  [LAMBDA (CLAUSE CPYFLG VAL)
    (PROG (OPR0 (CLTYP0 'COND)
          TEM1 TEM2 PARENT SUBPARENT)
```

(* CLTYP0 is bound inform CLISP3 that it is ok to remove parentheses from expressions converted to infix notation, e.g. (SETQ CLAUSE (FOO)))

```
          (RETURN (COND
                     [(AND VAL (EQ (CAR CLAUSE)
                                   T))
```

(* Don't use ELSE unless previous clauses seen, otherwise (COND (T --)) gets messed up, i.e. becomes IF --.)

```
                      (CONS (COND
                                (LCASEFLG 'else)
                                (T 'ELSE))
                            (CLISPCOND1 (CDR CLAUSE]
                     (T [SETQ TEM1 (AND (CDR CLAUSE)
                                        (CONS (COND
                                                  (LCASEFLG 'then)
                                                  (T 'THEN))
                                              (CLISPCOND1 (CDR CLAUSE)
                                                          CPYFLG]
                        (SETQ TEM2 (CLISPIFY2 (CAR CLAUSE)))
                        (CONS (COND
                                  (LCASEFLG 'elseif)
                                  (T 'ELSEIF))
                              (COND
                                  (SEG (SETQ SEG NIL)
                                       (NCONC TEM2 TEM1))
                                  ((CLREMPARS TEM2)          (* Says is a small list.)
                                   (APPEND TEM2 TEM1))
                                  (T (CONS TEM2 TEM1])
```

(CLISPCOND1

```
  [LAMBDA (L CPYFLG)
```

(* If CPYFLG is T, something will be NCONCed onto the value returned by CLISPCOND1, so we must make sure that it does not aapear in the original function.)

```
    (PROG (TEM)
          [SETQ TEM (CLISPIFY1 L NIL (AND (NULL (CDR L))
                                          'COND)
                               NIL
                               (OR NOVALFLG 'NOTLAST]
          (RETURN (COND
                     [(AND (NULL (CDR TEM))
                           (CLREMPARS (CAR TEM)))
                      (COND
                         ([AND CPYFLG (EQ (FLAST (CAR TEM))
                                          (FLAST (CAR L]
```

(* FLAST i necesaary because forms may not be EQ but still have common tails, ee.g. (FOO (SETQ X Y) Z) becomes (FOO X_Y Z)%, but (Z) is same as in original expression.)

```
                          (APPEND (CAR TEM)))
                         (T (CAR TEM]
                     ((AND CPYFLG (EQ (FLAST TEM)
                                      (FLAST L)))
                      (APPEND TEM))
                     (T TEM])
```

(CLISPAND

```
  [LAMBDA (FORM)                                        (* wt%: 3-AUG-77 3 1)
    (PROG (TEM)
```

```
          (CLISP4 FORM)
          (RETURN (CLISPCOND (COND
                                ((OR (NULL (CDDDR FORM))
                                     (CLISPAND1 (CADDR FORM)))

          (* E.G. (AND X Y) -> IF X THEN Y. Similary, (AND X Y --) -> IF X THEN Y --
          if it is known that Y is always true.)

                                 (CDR FORM))
                                (T                                    (* E.G. (AND X Y Z) -> IF X AND Y THEN z.)
                                   (CONS [LDIFF FORM (SETQ TEM (OR (SOME (CDDDR FORM)
                                                                         (FUNCTION CLISPAND1))
                                                                   (FLAST FORM]
                                         TEM])
```

(CLISPAND1

```
  [LAMBDA ($FORM)

          (* Returns T if $FORM is known to return a NON-NIL value. used in clispifying ANDs.)

    (COND
       ((LISTP $FORM)
        (SELECTQ (CAR $FORM)
             ((CONS LIST RPLACA RPLACD FRPLACA FRPLACD /RPLACA /RPLACD)
                 T)
             (QUOTE (CADR $FORM))
             (SETQ (CLISPAND1 (CADDR $FORM)))
             (SETQQ (CADDR $FORM))
             ((PRINT PRIN1)
                  (CLISPAND1 (CADR $FORM)))
             (COND [AND (CLISPAND1 (CAAR (FLAST $FORM)))
                        (EVERY (CDR $FORM)
                               (FUNCTION (LAMBDA (CLAUSE)
                                                 (CLISPAND1 (CAR (FLAST CLAUSE])
             NIL))
       ((LITATOM $FORM)
        (EQ $FORM T))
       (T T])
```

(CLISPIFYNOT

```
  [LAMBDA (FORM)                                                  (* lmm "12-AUG-84 23:28")
    (PROG (TEM1 TEM2)
          (SETQ NEGFLG (NOT NEGFLG))
          (SETQ TEM1 (CLISPIFY2B FORM))

          (* reason we dont want to call CLISPIFY2 is in some cses, the NEGFLG will be taken care of below, e.g.
          (NOT (ILESSP X Y)) goes to (X GEQ Y)%. in this case, NEGFLG is reset
          (in CLISP3)%, and CLISPIFY2 rebinds NEGFLG, so must call CLISPIFY2b instead)

          (RETURN (COND
                     ((NULL NEGFLG)
                      TEM1)
                     ([AND CLISPFLG (GETPROP 'NOT 'CLISPINFIX)
                           (SETQ TEM2 (GETPROP '~ 'CLISPTYPE]
                      (CLISP3 '~ (LIST '~ TEM1)
                              TEM2 T))
                     (T (LIST 'NOT TEM1])
```

(CLISPIFYMATCHUP

```
  [LAMBDA (PAT $LST $VARS ALST)                                   (* wt%: 13-FEB-76 20 29)

          (* like clispmatchup except also recurses down into lists, and distinguishes matches between elements and tails.
          clispmatchup doesnt have to do this)

    (PROG (TEM)
      LP  (COND
             ((NLISTP $LST)
              (RETURN NIL))
             [(FMEMB (CAR PAT)
                     $VARS)
              (COND
                 [(NOT (SETQ TEM (FASSOC (CAR PAT)
                                         ALST)))
                  (SETQ ALST (NCONC1 ALST (CONS (CAR PAT)
                                                (CAR $LST]
                 ((NOT (EQUAL (CDR TEM)
                              (CAR $LST)))                        (* e.g. if (X IS POSITIVE) is defined as
                                                                  (AND (NUMBERP X) (IGREATERP X 0)) then
                                                                  (AND (NUMBERP Y) (IGREATERP Z 0)) cant translate to
                                                                  (Y IS POSITIVE))
                  (RETURN NIL]
             ((EQ (CAR PAT)
                  (CAR $LST)))
             [(AND (LISTP (CAR PAT))
```

```
                (LISTP (CAR $LST)))
            (COND
               ((NULL (SETQ ALST (CLISPIFYMATCHUP (CAR PAT)
                                        (CAR $LST)
                                        $VARS ALST)))
                (RETURN NIL]
           ((EQ (CAR (GETPROP (CAR PAT)
                              'CLISPCLASS))
                'ISWORD)
            (SETQ PAT (CDR PAT))
            (GO LP))
           ((FMEMB (CAR PAT)
                   CLISPISNOISEWORDS)                    (* e.g. A, AN, THE etc.)
            (SETQ PAT (CDR PAT))
            (GO LP))
           [(EQ (CAR PAT)
                (GETPROP (CAR $LST)
                         'CLISPISPROP]
           (T (RETURN NIL)))
         (COND
           ((SETQ PAT (CDR PAT))
            (SETQ $LST (CDR $LST))
            (GO LP))
           ((NULL (CDR $LST))
            (RETURN (OR ALST T)))
           (T (RETURN NIL])
```

(CLREMPARS

```
[LAMBDA (X)
   (AND CLREMPARSFLG (LISTP X)
        (CDR X)
        (NULL (CDDDR X))
        (ATOM (CAR X))
        (ATOM (CADR X))
        (ATOM (CADDR X))
        (FGETD (CAR X))
        (NULL (STRPOSL CLISPCHARRAY (CAR X])
```

(CLISPIFYCROPS0

```
[LAMBDA (CROPSLST)                                       (* wt%: 3-AUG-77 3 1)
   (CLISP4 FORM)                                         (* Handles things like (CAR X_Y) and
                                                         (LAST X_Y) by first dwimifying,, when necessary.)
   (CLISPIFYCROPS (CADR FORM)
          CROPSLST
          (CAR FORM])
```

(CLISPIFYCROPS

```
[LAMBDA (X CROPSLST CROPFN Y)                            (* lmm "16-Aug-84 14:17")
```

(* X was originally of the form (car/cdr/... X)%. Y is given on calls from CLISPIFYRPLAC.
In this case, Y is either NCONC, NCONC1, etc. or CAR or CDR
(correspnding to RPLACA or RPLACD)%. Y tells CLISPIFYCROPS not to do a CLISP3A, and is also added to the end of the CROP operatrs.)

```
   (PROG (TEM1 TEM2 PARENT SUBPARENT (PARENT0 PARENT))
         [COND
            ([AND (SETQ TEM1 (OR (CAR CROPSLST)
                                 Y))
                  (COND
                     [(LITATOM TEM1)
                      (OR (FMEMB TEM1 CLISPCHARS)
                          (AND (FMEMB CROPFN '(fetch replace))
                               (STRPOS "." TEM1]
                     (T (AND (FMEMB CROPFN '(fetch replace))
                             (SOME TEM1 (FUNCTION (LAMBDA (TEM1)
                                                  (STRPOS "." TEM1]
             (RETEVAL 'CLISPIFY2B 'FORM))
            ([AND (NULL Y)
                  (OR (NULL CLISPFLG)
                      (NULL CL%:FLG)
                      (NULL (SETQ TEM1 (GETPROP '%: 'CLISPTYPE]  (* CLISPIFYRPLAC makes this check before calling
                                                                 CLISPIFYCROPS.)
             (RETURN (CLISPIFY2C FORM]
         (PROG (OPR0 CLTYP0)
               [COND
                  (CL%:FLG (SETQQ OPR0 %:)
                           (SETQ CLTYP0 (GETPROP '%: 'CLISPTYPE]
```

(* This means that if %: is encountered in the course of clipifying x, clisifycrops will just return the list of %:1's and |::1's.| Thus CADR of CDDR will clisify to %:4, not |::3:1.|)

```
               (SETQ TEM1 (CLISPIFY2 X)))
         (COND
            ((NULL SEG)                                  (* Makes rest of program simpler since TEM1 now always
                                                         corresponds to a segment.)
```

```
            (SETQ TEM1 (LIST TEM1)))
           ((OR (NULL CL%:FLG)
                (NEQ (CADR TEM1)
                     CAR/CDRSTRING))
```

(* The NEQ says TEM1 is a sequence of operators and operands other than a CAR or CDR perator, E.g. (CADR (SETQ X Y)))

```
            [COND
               ((CDR (SETQ TEM1 (CLISP3A TEM1)))
                (SETQ TEM1 (LIST TEM1))
```

(* This insures that the clispified form will be parenthesized. This is necessar unless it reduces to a single atom, i.e. CLISP3A returns a list of one element, since otherwise, the operator might be broadscope, e.g. (CADR (AND X Y)))

```
                ]
            (GO OUT)))
        (COND
           ([OR (NULL CL%:FLG)
                (COND
                   [(LISTP (CAR TEM1))
                    (COND
                       [(EQ CL%:FLG T)
                        (OR (CDDAR TEM1)
                            (LISTP (CADAR TEM1]
                       ((EQ CL%:FLG 'ALL)                    (* Says go back to %: notation regardless of length of expression.)
                        (SOME (CAR TEM1)
                              (FUNCTION LISTP)))
                       ((LISTP CL%:FLG)
                        (NULL (APPLY* CL%:FLG (CAR TEM1]
                   (T (CLISPNOEVAL (CAR TEM1]                (* E.G. The first operand is a list, therefore don't use %: notation.)
            (GO OUT)))
        (COND
           [(EQ OPR0 '%:)
```

(* Leaves it as A's and D's for higher operator, which is also a %:, to process. The reason for doing this is that (CAR (CDDR X)) can therefore become %:3 not |::2:1.|)

```
            (SETQ SEG T)
            [COND
               ((NEQ (CADR TEM1)
                     CAR/CDRSTRING)
```

(* Special STRING used to mark list to inidicate that what follows is a list of A's and D's for CLISPIFYCROPS1. Note that the marker may already be in there from a lower call to CLISPIFYCROPS)

```
                (SETQ CROPSLST (CONS CAR/CDRSTRING CROPSLST]
            (RETURN (NCONC TEM1 (APPEND CROPSLST]
           ((EQ (CADR TEM1)
                CAR/CDRSTRING)
            (FRPLACD TEM1 (CDDR TEM1))                       (* Reeove marker)
            (SETQ TEM2 TEM1))
           ((NULL (CDR TEM1))
```

(* Of form (atom)%, e.g. CLISPIFYCROPS was called with X an atom, as in CLISPIFYING (CAR X) (Remember that we hae listed the result of CLISPIFYING so that all cses can be treated as segments))

```
            (SETQ TEM2 TEM1))
           (T (SHOULDNT)))
        (FRPLACD TEM2 (CLISPIFYCROPS1 (NCONC (CDR TEM2)
                                             CROPSLST)
                              Y CROPFN))
        (SETQ SEG T)
        [RETURN (COND
                   ((CLISP3B '%: (GETPROP '%: 'CLISPTYPE))
                    (SETQ SEG NIL)
                    (SETQ PARENT PARENT0)
                    (CLISP3A TEM1))
                   ((OR Y OPR0)
                    TEM1)
                   (T (SETQ PARENT PARENT0)
                      (CLISP3A TEM1]
OUT                                                          (* dont use %: notation)
        (SETQ SEG NIL)
        (RETURN (COND
                   ((NULL Y)
```

(* TEM1 is the CLISPIFIED X, as a segment, so CONS the CROPFN back on it.)

```
                    (SELECTQ CROPFN
                         (fetch (CONS 'fetch (CONS (CAR CROPSLST)
                                                   (CONS 'of TEM1))))
                         (match (CONS CROPFN (APPEND TEM1 (CONS 'with CROPSLST))))
                         (CONS CROPFN TEM1)))
```

```
                    ((NULL CROPFN)
                     TEM1)
                    (T (LIST (CONS CROPFN TEM1])
```

(CLISPIFYCROPS1

```
[LAMBDA ($LST Y CROPFN)                                    (* wt%: 27-JUN-77 23 41)
```

(* takes a list consisting of %:1 (for car) |::1| (for cdr)%, numbers (for nth)%, other litatoms (from record operations)%, and lists (from pattern matches or records) and produces the appropriate list contaiing just %:'s and numbers suitable for packing.)

```
  (PROG (X N TAILSTATE TEM)
    LP  (COND
           ($LST (SETQ TEM (CAR $LST)))
           (Y (SETQ TEM Y)
              (SETQ Y NIL))
           (T (SETQ TEM NIL)
              (GO OUT)))
    LP1 [SELECTQ TEM
          (%:1
```

(* %:1 used instead of CAR, or A, because want to choose a name that is unlikely to appear as a record field.)

```
                (SETQ X (CONS (COND
                                 ((NULL TAILSTATE)
                                  1)
                                 ((MINUSP N)
                                  N)
                                 (T (ADD1 N)))
                              (CONS '%: X)))
                (SETQ TAILSTATE NIL))
           (|::1| (COND
                     ((NULL TAILSTATE)
                      (SETQ TAILSTATE T)
                      (SETQ N 1))
                     ((NEQ N -1)
                      (SETQ N (ADD1 N)))
                     (T [SETQ X (CONS -1 (CONS '%: (CONS '%: X]
                        (SETQ N 1))))
           (COND
              [(NUMBERP TEM)
               (COND
                  ((NULL TAILSTATE)
                   (SETQ TAILSTATE T)
                   (SETQ N TEM))
                  ((IGREATERP TEM 0)
                   (SETQ N (IPLUS TEM N)))
                  (T                                       (* e.g. (LAST (CDR x)))
                     [SETQ X (CONS N (CONS '%: (CONS '%: X]
                     (SETQ N TEM]
              (T [AND TAILSTATE (SETQ X (CONS N (CONS '%: (CONS '%: X]
                 (SETQ TAILSTATE NIL)
                 (COND
                    [(NLISTP TEM)                          (* ACCESS function, e.g. X%:FOO.)
                     (SETQ X (CONS TEM (CONS '%: X]
                    [(EQ CROPFN 'match)
                     (SETQ X (CONS TEM (CONS '%: X]
                    [(CDR TEM)                             (* access path, e.g. (FETCH (A B) OF C))
                     (SETQ X (CONS [PACK (CONS (CAR TEM)
                                               (MAPCONC (CDR TEM)
                                                      (FUNCTION (LAMBDA (X)
                                                                  (LIST '%. X]
                                   (CONS '%: X]
                    (T                                     (* (FETCH (FOO) OF FIE) same as
                                                           (FETCH FOO OF FIE))
                       (SETQ X (CONS (CAR TEM)
                                     (CONS '%: X]
           (SETQ $LST (CDR $LST))
           (GO LP)
    OUT [AND TAILSTATE (SETQ X (CONS N (CONS '%: (CONS '%: X]
           (AND TEM (SETQ X (APPEND TEM X)))               (* Adds the %: or |::| foo NCONC or Nconc2)
           (RETURN (DREVERSE X])
```

(CLISPIFYRPLAC

```
[LAMBDA (X TYP %:FLG)                                      (* wt%: 3-AUG-77 3 1)
  (PROG (TEM CROPS (CLTYP00 CLTYP0)
             (OPR0 '_)
             (CLTYP0 (GETPROP '_ 'CLISPTYPE))
             LFT RGHT TYP0 (PARENT0 PARENT)
             PARENT SUBPARENT)
        [COND
           ([OR (NULL CLTYP0)
                (NULL CLISPFLG)
                (NULL CL%:FLG)
                (NULL (GETPROP '%: 'CLISPTYPE]             (* _ transformation disabled)
```

```
            (RETURN (CLISPIFY2C X]
         (CLISP4 X)

         (* To handle cases like (RPLACA X_Y T)%, which if not first dwiified, would go to X_Y%:1_T.)

         (COND
            ((EQ TYP 'replace)
             (SETQ LFT (CADDDR X))
             (SETQ RGHT (CDR (CDDDDR X)))
             (SETQ TYP0 (CADR X)))
            ((NEQ (CAR X)
                  (CLISPIFYLOOKUP (CAR X)
                         (CADR X)
                         (CADDR X)))                    (* E.g. RPLACA-RPLACD being used in this function and this is
             an FRPLACA.)
             (RETURN (CLISPIFY2C X)))
            (T (SETQ LFT (CADR X))
               [SETQ RGHT (NTH (CDR X)
                               (OR (GETPROP (CAR X)
                                            'NARGS)
                                   (AND (NOT (SUBRP (CAR X)))
                                        (NARGS (CAR X)))
                                   (PUTPROP (CAR X)
                                            'NARGS
                                            (LENGTH (SMARTARGLIST (CAR X]

         (* The problem is finding which of the arguments in thise xpression belong to the accessfunction and which to the setfn, e.g.
         Can't just default to all but last because last might not be supplied, e.g.
         (RPLACA X) must clispify to X%:1_NIL, not X%:1_X. The number of arguments is obtained either from the property
         NARGS, of from the function NARGS, (if the function in question is not a SUBR//) or else an eror is gnerated)

               (SETQ TYP0 TYP)))
         [COND
            ((NULL %:FLG)

         (* Doesnt involve %:'s, e.g. from a SETFN. For example, if the original form were
         (SETA X Y Z)%, TYP would be ELT, and CLISPIFY2A would be called on
         (ELT X Y))

             [SETQ TEM (CLISPIFY2A (CONS TYP (LDIFF (CDR X)
                                                    RGHT]
             (GO OUT))
            ([AND (LISTP LFT)
                  (COND
                     ((SETQ CROPS (GETPROP (CAR LFT)
                                           'CROPS))
                      (SETQ CROPS (SUBPAIR '(A D)
                                           '(%:1 |::1|)
                                           CROPS]           (* E.g. (RPLACA (CDR X) --) becomes X%:2_--.
                                                            instead of |X::1:1_| --)
             (SETQ TEM (CLISPIFYCROPS (CADR LFT)
                              CROPS
                              (CAR LFT)
                              TYP0)))
            (T (SETQ TEM (CLISPIFYCROPS LFT NIL NIL TYP0]
         [COND
            ((NULL (CDR TEM))

         (* The first argument did not clispify to something containing %:'s, so we will not use the _ notation)

             (SETQ SEG NIL)
             (RETURN (COND
                        [(EQ TYP 'replace)
                         (CONS TYP (CONS TYP0 (CONS 'of (NCONC TEM (CONS 'with (CLISPIFY1 RGHT]
                        (T (CONS (CAR X)
                                 (NCONC TEM (CLISPIFY1 RGHT]
      OUT (SETQ SEG (NULL CLTYP00))
             (SETQ PARENT PARENT0)
             (RETURN (CLISP3A (NCONC TEM (COND
                                           ((CAR RGHT)
                                            (CONS '_ ([LAMBDA (LST TAIL)

         (* LST is rbound to T to indicate to CLISP3 that there are operands to the left of this expression
         (namely TEM and _)%. Otherwise things like (RPLACD X (OR (FOO Y) Y)) would go to |X::1_|
         (FOO Y) OR Y where actually the or should be parentheseized.
         TAIL is rebound to NIL so that CLISP3 will know there isnt anything on the right..)

                                                      (CLISPIFY2A (CAR RGHT]
                                                     T)))
                                           (T (LIST '_ NIL])
```

(CLISPIFYMAPS

```
  [LAMBDA (IN-ON OPR)                                   (* lmm "12-JUN-81 07:13")
    (PROG (VAR (FN1 (CADDR FORM))
               (FN2 (CADDDR FORM))
               TEM)
```

```
          (COND
             ([OR (NLISTP FN1)
                  (NEQ (CAR FN1)
                       'FUNCTION)
                  (AND (LISTP (SETQ FN1 (CADR FN1)))
                       (CDADR FN1))
                  [AND FN2 (OR (NLISTP FN2)
                               (NEQ (CAR FN2)
                                    'FUNCTION)
                               (AND (LISTP (SETQ FN2 (CADR FN2)))
                                    (CDADR FN2]
                  (NEQ (CAR FORM)
                       (CLISPIFYLOOKUP (CAR FORM)
                             (CADR FORM]                    (* E.G. (MAPCAR X Y))
              (RETURN NIL)))
          [SETQ VAR (COND
                       ((LISTP FN1)
                        (CAADR FN1))
                       ((EQ (CADR FORM)
                            'X)
                        'Y)
                       (T 'X]
          (COND
             ([AND (EQ OPR 'subset)
                   (OR (CDDDR (LISTP FN1))
                       (EDITFINDP FN1 (LIST 'SETQ VAR '--]
              (RETURN NIL)))
          (RETURN (NCONC (AND VAR (LIST (COND
                                           (LCASEFLG 'for)
                                           (T 'FOR))
                                        VAR))
                         (LIST IN-ON)
                         (COND
                            ((AND [NULL (CDR (SETQ TEM (CLISPIFY2A (CADR FORM]
                                  (CLREMPARS (CAR TEM)))
                             (APPEND (CAR TEM)))
                            (T TEM))
                         (CLMAPS2 FN2 (COND
                                        (LCASEFLG 'by)
                                        (T 'BY))
                                  VAR)
                         (CLMAPS2 FN1 OPR VAR])
```

(CLMAPS1

```
  [LAMBDA (FN)                                              (* wt%: 13-FEB-76 19 40)
    (COND
       ((NEQ (CAR FN)
             'F/L)
        (CADR FN))
       (T (CONS 'LAMBDA (COND
                          ((AND (CDDR FN)
                                (NOT (FGETD (CAADR FN)))
                                (EVERY (CADR FN)
                                       (FUNCTION ATOM)))
                           (CDR FN))
                          (T (CONS (LIST 'X)
                                   (CDR FN])
```

(CLMAPS2

```
  [LAMBDA (DEF WORD VAR)                                    (* lmm "12-JUN-81 07:17")
    (AND DEF (PROG (X Y TEM OPR0 CLTYP0)
                   [COND
                      ((EQ WORD 'subset)
                       [SETQ WORD (COND
                                     (LCASEFLG 'when)
                                     (T 'WHEN]
                       (SETQ Y (LIST (COND
                                        (LCASEFLG 'collect)
                                        (T 'COLLECT))
                                     VAR]
```

(* The expression constructed by clmaps2 is of the form WORD body when/unless pred.
body corresponds to the functional argument. In the case of subset, it is when/unless body collect var.)

```
                   [SETQ X (COND
                              [(NLISTP DEF)
                               (COND
                                  ((FNTYP DEF)
                                   (LIST DEF))
                                  (T                        (* or otherwise wont dwimify back right)
                                     (LIST (LIST DEF VAR]
                              ([AND (FMEMB WORD '(DO JOIN do join))
                                    (NULL (CDDDR DEF))
                                    (COND
                                       ((AND (EQ (CAR (SETQ X (CADDR DEF)))
```

```
                                        'COND)
                                   (NULL (CDDR X)))    (* The form of the function is (LAMBDA &
                                                       (COND (--))) TEM is set to the clause.)
                              (SETQ TEM (CADR X)))
                             ((EQ (CAR X)
                                  'AND)                (* If the NULL yields true, the form is
                                                       (AND & &))
                              (NULL (CDDR (SETQ TEM (CDR X]
                        (SETQ Y TEM)
                        (SETQ X (CDR Y))              (* X now corresonds to the body of the iteraion.)
                        [SETQ Y (CONS [COND
                                         [(EQ (CAAR Y)
                                              'NOT)
                                          (SETQ Y (CADAR Y))
                                          (COND
                                             (LCASEFLG 'unless)
                                             (T 'UNLESS]
                                         (T (SETQ Y (CAR Y))
                                            (COND
                                               (LCASEFLG 'when)
                                               (T 'WHEN]
                                      (COND
                                         ((AND [NULL (CDR (SETQ TEM (CLISPIFY2A Y]
                                               (CLREMPARS (CAR TEM)))
                                          (CAR TEM))
                                         (T TEM]
                        [COND
                           ((AND (OR (EQ WORD 'JOIN)
                                     (EQ WORD 'join))
                                 [NULL (CDDAR (SETQ TEM (FLAST X]
                                 (EQ (CAAR TEM)
                                     'LIST))
                            [SETQ WORD (COND
                                          (LCASEFLG 'collect)
                                          (T 'COLLECT]
                            (SETQ X (NCONC (LDIFF X TEM)
                                           (CDAR TEM]     (* E.g. JOIN (COND (& -- (LIST &))) -> COLLECT --
                                                          WHEN &)
                        (CLISPIFY1 X))
                       (T (CLISPIFY1 (CDDR DEF]
                [COND
                   ((AND (LISTP DEF)
                         (OR (CDADR DEF)
                             (NEQ (CAADR DEF)
                                  VAR)))
```

(* Entire LAMBDA expression must be included because the variable is not the same as that in the FOR, i.e. not the same as the one in the first functional argument.)

```
                    (RETURN (CONS WORD (LIST (CONS (CONS (CAR DEF)
                                                         (CONS (CADR DEF)
                                                               X))
                                                   Y)
                                             VAR]
                (RETURN (CONS WORD (COND
                                      ((AND (NULL (CDR X))
                                            (CLREMPARS (CAR X)))
                                       (APPEND (CAR X)
                                               Y))
                                      (T (APPEND X Y]))
```

(CLSTOPSCAN?

```
[LAMBDA (CLTYPX CLTYP)
```

(* STOPSCAN? is T if operator corresponding to CLTYPX would stop scan for operator corresponding to CLTYP, i.e. if former is of lower or same precedence as latter.)

```
   (AND CLTYPX CLTYP (NOT (ILESSP (COND
                                     ((ATOM CLTYP)
                                      CLTYP)
                                     (T (CDR CLTYP)))
                                  (COND
                                     ((ATOM CLTYPX)
                                      CLTYPX)
                                     (T (CAR CLTYPX])
```

(CLISPIFYLOOKUP

```
[LAMBDA (WORD VAR1 VAR2 CLASS CLASSDEF)                    (* wt%: 31-MAY-76 22 34)
```

(* In most cases, it is not necessary to do a full lookup. This is q uick an dirty check inside of the block to avoid calling CLISPLOOKUP0 whenever there are no declarations.)

```
   (PROG (TEM)
         [OR CLASS (SETQ CLASS (GETPROP WORD 'CLISPCLASS]
         [OR CLASSDEF (SETQ CLASSDEF (GETPROP CLASS 'CLISPCLASSDEF]
```

```
          [SETQ TEM (COND
                       ((AND CLASS DECLST)
```

(* must do full lookup. Note that for CLISPLOOKUP, CLISPLOOKUP0 is only called when there is a CLASSDEF. Here it is called when there is a CLASS property. This is bcause what CLISPIFYLOOKUP is really asking is what would the infix operator corresponding to WORD go to if DWIMIIED, e.g. if WORD is FGTP, CLISPIFYLOOKUP is reaally asking what does GT go to.)

```
                        (CLISPLOOKUP0 WORD VAR1 VAR2 DECLST NIL CLASS CLASSDEF))
                       (T
```

(* The last GETPROP %, i.e. for CLASS, is so we dont have to implement global declaraions by puttig a LISPFN property on each member of the class.)

```
                          (OR (GETPROP WORD 'LISPFN)
                              (GETPROP CLASS 'LISPFN)
                              WORD]
          [COND
             ((AND (EQ (CAR CLASSDEF)
                       'ARITH)
                   (EQ TEM (CADR CLASSDEF))
                   (OR (FLOATP VAR1)
                       (FLOATP VAR2)))
              (SETQ TEM (CADDR CLASSDEF]
          (RETURN TEM])
```

(**LOWERCASE**

```
  [LAMBDA (FLG)                                                    (* wt%: 13-FEB-76 19 40)
    (PROG1 LCASEFLG
        [PROG (FN TEM)
              (AND (NULL CHCONLST)
                   (SETQ CHCONLST 'NIL))                           (* Because LOWERCASE is often done in initialization, i.e.
                                                                   before CHCONLST is set.)
              [SETQ FN (COND
                          (FLG 'L-CASE)
                          (T 'U-CASE]
              (RPAQ LCASEFLG FLG)
              [MAPC '(MAPC MAP MAPCAR MAPLIST MAPCONC MAPCON)
                    (FUNCTION (LAMBDA (X)
                                (/PUT X 'CLMAPS (CONS [APPLY* FN (CAR (SETQ TEM (GETPROP X 'CLMAPS]
                                                      (APPLY* FN (CDR TEM]
              (/PUT 'OR 'CLISPINFIX (APPLY* FN 'OR))
              (/PUT 'AND 'CLISPINFIX (APPLY* FN 'AND])])
```

(**SHRIEKIFY**

```
  [LAMBDA (LOOKAT)                                                 (* wt%: "23-JUL-78 23:31")
    (PROG (RESULTP CARTEST (OPR0 '<)
           (CLTYP0 (GETPROP '< 'CLISPTYPE))
           PARENT SUBPARENT)
          (SELECTQ (SETQ CARTEST (CAR LOOKAT))
              ((NCONC /NCONC)
                   (COND
                      ((NEQ CARTEST (CLISPIFYLOOKUP 'NCONC NIL))
                       (RETURN NIL))))
              ((NCONC1 /NCONC1)
                   (COND
                      ((NEQ CARTEST (CLISPIFYLOOKUP 'NCONC1 NIL))
                       (RETURN NIL))))
              NIL)
          (CLISP4 FORM)
          (SETQ RESULTP (LIST '<))
          (SHRKFY LOOKAT 'STARTING T)
          [SETQ RESULTP (COND
                           ((CDR RESULTP)
                            (NCONC1 RESULTP '>]
          (RETURN (COND
                     ([AND (LITATOM (CADR RESULTP))
                           (FNTYP (CADR RESULTP))
                           (NOT (BOUNDP (CADR RESULTP]
                      NIL)
                     ([OR (AND (NULL (GETPROP '! 'CLISPWORD))
                               (EDITFINDP RESULTP '!))
                          (AND (NULL (GETPROP '!! 'CLISPWORD))
                               (EDITFINDP RESULTP '!!]
```

(* e.g., rich fikes likes to disable ! without disabling <. this is not the most efficient way to make the check, but there are somany places in shrkfy where !'s are put in, is easier to justcheck after wards.)

```
                      NIL)
                     (T RESULTP])
```

(**SHRKFY**

```
  [LAMBDA (LOOKAT WORKFLAG STAGEFLAG)                              (* wt%: "23-JUL-78 23:30")
```

(* SHRKFY is a translator from LISP expressions involving CONS, LIST, APPEND, NCONC, NCONC1, /NCONC, and /NCONC1 to CLISP expressions using !, !!, and <; thus it is the inverse translator to SHRIEKER.
Although this is a large program, its operation is fairly simple. Several prog labels, from A1 to A5, have been introduced to aid this explication. Control flows straight through SHRKFY, from top to bottom, with no awkward detours or loops.
In essence, SHRKFY is a stack of three large selectq's, each of which does some computation necessary for the next.)

```
(PROG ((CARSAFEFLAG T)
       CARLOOKAT CDRLOOKAT CDARLOOKAT CAARLOOKAT OPFLG CARFLAG CDDARLOOKAT RESULTQ RESULTR OP2FLG OP3FLG
       FIRSTARGFLG APPSINGFLG)
      (SETQ CARLOOKAT (CAR LOOKAT))
      (COND
         ((LISTP CARLOOKAT)
          (SETQ CAARLOOKAT (CAR CARLOOKAT))
          (SETQ CDARLOOKAT (CDR CARLOOKAT))
          (SETQ CDDARLOOKAT (CDR CDARLOOKAT))
          (SETQQ CARFLAG ITSALIST))
         (T (SETQQ CARFLAG ELEMENTAL)))
```

(* These canonical prog varnames remain constant throughout the program.
I.e. CAARLOOKAT is always (CAAR LOOKAT)%, etc.)

```
      (SETQ CDRLOOKAT (CDR LOOKAT))
  A1
```

(* SHRKFY works by emulating, or mimicing, the actions of APPEND, CONS, LIST, NCONC, NCONC1, etc., on their arguments with respect to the CLISP operators !, !!, and <. Whenever SHRKFY is called, WORKFLAG is the name of the function being emulated and STAGEFLAG is the "stage" (either T or NIL) that the emulation has reached.
The first time that SHRKFY is called to mimic a function, STAGEFLAG will be T, which is SHRKFY's signal that this is indeed the first time it has been called, and that LOOKAT is CDR of the original form.
STAGEFLAG will then be setq'd to NIL. Depending on the value of STAGEFLAG and CDRLOOKAT (which tells SHRKFY whether or not there or more arguments besides CARLOOKAT)%, OPFLG will be setq'd to %'!, %'!!, or %'LISTIT, and control will then flow to A2.)

```
      (SELECTQ WORKFLAG
           ((NCONC /NCONC)
                (COND
                   (STAGEFLAG (SETQQ OPFLG !!))
                   (CDRLOOKAT (SETQQ OPFLG !!))
                   (T (SETQQ OPFLG !)))
                (COND
                   (STAGEFLAG
```

(* FIRSTARGFLG is setq'd to T to save the fact that CARLOOKAT is the first argument of the form.)

```
                            (SETQ STAGEFLAG NIL)
                            (SETQ FIRSTARGFLG T))))
           (CONS (COND
                    (STAGEFLAG (SETQQ OPFLG LISTIT)
                               (SETQ STAGEFLAG NIL)
                               (SETQ FIRSTARGFLG T))
                    (CDRLOOKAT (SETQQ OPFLG LISTIT))
                    (T (SETQQ OPFLG !))))
           (APPEND (COND
                      (STAGEFLAG (SETQ STAGEFLAG NIL)
                                 (SETQ FIRSTARGFLG T)))
                   (SETQQ OPFLG !))
           (LIST (SETQQ OPFLG LISTIT))
           ((NCONC1 /NCONC1)
                (COND
                   (STAGEFLAG (SETQQ OPFLG !!))
                   (T (SETQQ OPFLG LISTIT)))
                (COND
                   (STAGEFLAG (SETQ STAGEFLAG NIL)
                              (SETQ FIRSTARGFLG T))))
           (STARTING
```

(* The very first time that SHRKFY is called (by SHRIEKIFY)%, WORKFLAG is eq to %'STARTING.
This branch takes care of recognizing whether the form LOOKAT has at least one argument.
If it does, then SHRKFY is called recursively on CDRLOOKAT, with WORKFLAG = CARLOOKAT.
Otherwise SHRKFY returns to SHRIEKIFY.)

```
                  (COND
                     (CDRLOOKAT                          (* the form has at least one argument.)
                            (RETURN (SHRKFY CDRLOOKAT CARLOOKAT STAGEFLAG)))
                     (T (SELECTQ CARLOOKAT
                             ((LIST APPEND NCONC /NCONC)
```

(* (APPEND)%, (LIST)%, (NCONC)%, and (/NCONC) all evaluate to NIL.
RESULTP will be (<) when we return to SHRIEKIFY, which will return NIL.)

```
                                 (RETURN NIL))
                             ((CONS NCONC1 /NCONC1)
```

(* (CONS)%, (LIST)%, (NCONC1)%, (/NCONC1) all evaluate to (NIL)%, so this branch adds NIL to RESULTP and returns to SHRIEKIFY.)

```
                                 (RETURN (NCONC1 RESULTP NIL)))
                             NIL))))
```

```
            NIL)
      A2  [COND
             (CDRLOOKAT
```

(* RESULTR holds SHRKFY's translation of the arguments after CARLOOKAT.
Nothing will be done with it until the final COND at the top level of the SHRKFY prog, which takes care of adding RESULTR onto RESULTP. The next three selectq's at A3, A4, and A5 are devoted to adding the proper translation of CARLOOKAT to RESULTP.)

```
                    (SETQ RESULTR (SHRKFY2 CDRLOOKAT WORKFLAG STAGEFLAG))
                    (SETQ RESULTR (CDR RESULTR]
      A3  (SELECTQ OPFLG
                ((!! !)
                     (SELECTQ CARFLAG
                          (ELEMENTAL [COND
                                        (CARLOOKAT
```

(* If CARLOOKAT is not nil and not a list then we just add it on to RESULTP, preceded by the appropriate operator (%'! or %'!!)%. The selectq with the call to DWIMIFY1A enables us to catch errors like (APPEND A B CONS D E) and issue a message to the user that there is a "(possible) parentheses error." SHRKFY, however, continues with its computation.)

```
                                              (NCONC RESULTP (LIST OPFLG CARLOOKAT))
                                              (SELECTQ CARLOOKAT
                                                   ((APPEND CONS LIST NCONC NCONC1 /NCONC /NCONC1 QUOTE)
                                                        (DWIMIFY1A FORM LOOKAT CLISPIFYFN))
                                                   NIL))
                                        (T
```

(* makes sure that (APPEND NIL A)%, (NCONC NIL A)%, etc. go to <! NIL ! A>, <!! NIL ! A>, not <! A >.
Otherwise, ! NIL and !! NIL are left out of RESULTP. Thus, (APPEND A B NIL C D) goes to <! A ! B ! C ! D>.
This conditional could be refined a little to let cases like (NCONC1 NIL A) go to < A >, rather than <!! NIL A>.)

```
                                         (COND
                                            ((AND FIRSTARGFLG (NULL (CDR CDRLOOKAT))
                                                  CDRLOOKAT)
                                             (NCONC RESULTP (LIST OPFLG NIL])
                    (ITSALIST                                    (* CARLOOKAT is a list (form)%.)
                              [COND
                                 [CDARLOOKAT
```

(* If CDARLOOKAT is non-nil then we know there's at least one argument in the form, so we do a selectq on CAARLOOKAT, the first element of the form, which is expected to be a function name.
This selectq finds out which function name, and saves this information in OP2FLG.
(In certain cases, CARSAFEFLAG will be setq'd to NIL.) Without exception, control then flows to the major selectq on OP2FLG, which has the prog label A4.)

```
                                        (SELECTQ CAARLOOKAT
                                             ((LIST CONS)
                                                  (SETQ OP2FLG CAARLOOKAT))
                                             (APPEND (COND
                                                        ((NULL CDDARLOOKAT)
```

(* If CDDARLOOKAT is nil, then we know that the form CARLOOKAT, which has APPEND as its function name, has exactly one argument. So APPSINGFLG is setq'd to T, to save the fact that CARLOOKAT is an APPEND singleton.)

```
                                                            (SETQ APPSINGFLG T)))
                                                      (SETQQ OP2FLG APPEND))
                                             ((NCONC NCONC1 /NCONC /NCONC1)
                                                  (SETQ OP2FLG CAARLOOKAT)
```

(* CARSAFEFLAG is setq'd to NIL to indicate that CARLOOKAT may be
(in this case, is) a destructive operation.)

```
                                                  (SETQ CARSAFEFLAG NIL))
                                             (QUOTE
```

(* SHRKFY understands that if CARLOOKAT is a QUOTE form then it is not a destructive operation.
So CARSAFEFLAG is not affected, but OP2FLG is setq'd to OPFLG, which will result in calling CLISPIFY2A on CARLOOKAT.)

```
                                                    (SETQ OP2FLG OPFLG))
                                             (PROGN
```

(* CARLOOKAT is a form, and its first element is a function name that SHRKFY doesn't recognize.
So CARSAFEFLAG is setq'd to NIL, to indicate that there may be a destructive operation going on, and OP2FLG is setq'd to OPFLG (i.e. either %'! or %'!!)%, which will result in calling CLISPIFY2A on CARLOOKAT, when control flows to the selectq following the prog label A4.)

```
                                                    (SETQ OP2FLG OPFLG)
                                                    (SETQ CARSAFEFLAG NIL]
                                        (T
```

(* this branch handles ! (APPEND)%, !! (CONS)%, ! (CONS)%, !!
(NCONC)%, etc. I.e. CARLOOKAT is a form with no arguments.
If its function name is recognized by SHRKFY, then the appropriate code will be added automatically to RESULTP.
Although control will flow to the SELECTQ following A4, nothing will happen there, because OP2FLG is NIL.
Similarly for the SELECTQ on OP3FLG, following A5. Control will wind up at the final COND at the top level of the SHRKFY

prog, which takes care of adding RESULTR to RESULTP. On the other hand, if SHRKFY does not recognize the function name in CARLOOKAT, OP2FLG will be setq'd to OPFLG, which will cause CLISPIFY2A to be called on CARLOOKAT, when control flows to the selectq following A4.)

```
                                    (SELECTQ CAARLOOKAT
                                         ((APPEND NCONC LIST QUOTE /NCONC)
```

(* (APPEND) (NCONC) (LIST) (QUOTE) and (/NCONC) all evaluate to NIL.
Thus ! (APPEND) is the same as ! NIL, and can be left out of RESULTP, unless doing so would cause the next element in LOOKAT to be copied when it shouldn't be. E.g. (APPEND (APPEND) A) should go to
(<! NIL ! A>)%, not (<! A>)%. The same conditional is used to avoid this special case as in the branch above when CARFLAG = %'ELEMENTAL. This conditional could be refined a little to let cases like
(NCONC1 (APPEND) A) go to (< A >)%, rather than to (<!! NIL A>)%, as they do currently.)

```
                                                [COND
                                                   ((AND FIRSTARGFLG (NULL (CDR CDRLOOKAT))
                                                         CDRLOOKAT)
                                                    (NCONC RESULTP (LIST OPFLG NIL])
                                               ((CONS NCONC1 /NCONC1)
```

(* (CONS) (NCONC1) and (/NCONC1) all evaluate to (NIL)%, so this branch replaces !
(CONS) by ! <NIL>, etc. The brackets are left in for the sake of simplicity, because some cases require that they stay in. Thus if (NCONC A (CONS) B C) went to <!! A NIL !! B ! C>, then it would dwimify back to
(NCONC A (CONS NIL (NCONC B C)))%, which is not equivalent.
However, brackets can probably be left out whenever OPFLG = %'! and WORKFLAG = %'APPEND or %'CONS, which is a refinement that merits investigation. A small COND here would thus allow SHRKFY to simplify
(CONS A (APPEND)) to (LIST A NIL) and (APPEND (APPEND) A) to
(CONS NIL A)%.)

```
                                                   (NCONC RESULTP (LIST OPFLG '< NIL '>)))
                                              (PROGN
```

(* SHRKFY doesn't recognize the function name in CARLOOKAT, so this form will be given to CLISPIFY2A when control flows to the selectq following A4, and CARSAFEFLAG will be setq'd to NIL, to indicate that something destructive could be happening.)

```
                                                 (SETQ OP2FLG OPFLG)
                                                 (SETQ CARSAFEFLAG NIL])
                    NIL))
          (LISTIT
```

(* This branch is analogous to the one above (where OPFLG = %'! or %'!!)%, except that here CARLOOKAT is simply being listed, or added on.)

```
                   (SELECTQ CARFLAG
                        (ELEMENTAL
```

(* Note that there is an additional call to DWIMIFY1A here, which lets us catch errors like
(CONS NCONC D E) and issue a message to the user that there is a %'
(possible) parentheses error.)

```
                                    (NCONC1 RESULTP CARLOOKAT)
                                    (SELECTQ CARLOOKAT
                                         ((APPEND CONS LIST NCONC NCONC1 /NCONC /NCONC1 QUOTE)
                                             (DWIMIFY1A FORM LOOKAT CLISPIFYFN))
                                          NIL))
                         (ITSALIST [COND
                                      [CDARLOOKAT (SELECTQ CAARLOOKAT
                                                           ((CONS LIST APPEND)
                                                                (SETQ OP2FLG CAARLOOKAT))
                                                           ((NCONC NCONC1 /NCONC /NCONC1)
                                                                (SETQ CARSAFEFLAG NIL)
                                                                (SETQ OP2FLG CAARLOOKAT))
                                                           (QUOTE (SETQQ OP2FLG ADDITON))
                                                           (PROGN (SETQ CARSAFEFLAG NIL)
                                                                  (SETQQ OP2FLG ADDITON]
                                      (T (SELECTQ CAARLOOKAT
                                              ((APPEND NCONC LIST QUOTE /NCONC)
                                                   (NCONC1 RESULTP NIL))
                                              ((CONS NCONC1 /NCONC1)
                                                   (NCONC RESULTP (LIST '< NIL '>)))
                                              (PROGN (SETQQ OP2FLG ADDITON)
                                                     (SETQ CARSAFEFLAG NIL])
                         NIL))
          NIL)
A4   (SELECTQ OP2FLG
          ((!! !)
               (NCONC (NCONC1 RESULTP OP2FLG)
                      (CLISPIFY2A CARLOOKAT)))
          (ADDITON (NCONC RESULTP (CLISPIFY2A CARLOOKAT)))
          ((APPEND CONS LIST NCONC NCONC1 /NCONC /NCONC1)
```

(* CARLOOKAT is a form of at least one argument, and its function name is one of the special functions recognized by SHRKFY. This function name is the value of OP2FLG. Most of the general optimizations described in the memo on SHRKFY take place in this selectq.)

```
               (SELECTQ OP2FLG
```

```
                ((NCONC /NCONC)
```

(* If OP2FLG = %'NCONC, %'/NCONC, %'NCONC1, or %'/NCONC1, and is not eq to the value of the corresponding CLISPIFYLOOKUP, then control will be sent to A5, where CARLOOKAT will be given to CLISPIFY2C.)

```
                     (COND
                        ((NEQ OP2FLG (CLISPIFYLOOKUP 'NCONC NIL))
                         (SETQQ OP3FLG CLISPIFY2CIT)
                         (GO A5))))
                ((NCONC1 /NCONC1)
                     (COND
                        ((NEQ OP2FLG (CLISPIFYLOOKUP 'NCONC1 NIL))
                         (SETQQ OP3FLG CLISPIFY2CIT)
                         (GO A5))))
                NIL)
```

(* Within the prog below, FORM is rebound to CARLOOKAT, so that Warren's scanner will be appropriately triggered. SHRKFY2 is called, rather than SHRKFY, so that the lower level SHRKFY will be able to work with its own, fresh, RESULTP. The RESULTP that is returned by SHRKFY2 will be made the value of the current SHRKFY's RESULTQ. This RESULTQ will be a list of CLISP expressions, including ! and !!, without enclosing angle brackets.
The question of whether to add the angle brackets or not is resolved by the body of this branch, and the nature of this decision is stored in OP3FLG. Control then flows to the selectq following A5, where RESULTQ, with appropriate surrounding brackets (and preceding operators ! or !!)%, will be added to RESULTP.
We may think of RESULTQ as always, implicitly, having angle brackets around it, and thus the simple operation (NCONC RESULTP RESULTQ) corresponds to "removing" the angle brackets.
This operation is denoted by (SETQQ OP3FLG OFFANGLES)%, while the operation of leaving the brackets in and preceding them by %'!! or %'! is denoted by (SETQQ OP3FLG OPANGLE)%.)

```
          (PROG ((FORM CARLOOKAT))
                (SETQ RESULTQ (SHRKFY2 CDARLOOKAT OP2FLG T)))
          (SETQ RESULTQ (CDR RESULTQ))
          (COND
             (RESULTQ (SELECTQ OPFLG
                           ((!! !)
                                (COND
                                   [CDRLOOKAT (COND
                                                 ((AND CARSAFEFLAG (EQ WORKFLAG 'APPEND))
```

(* APPEND is the only non-destructive function which has OPFLG = %'! when CDRLOOKAT is non-nil.
By convention, brackets are never removed from RESULTQ when OPFLG = %'!!, nor are they ever removed when RESULTQ is "unsafe" (e.g. when RESULTQ contains %'!! at its top level) and CDRLOOKAT is non-nil; CARSAFEFLAG is nil if RESULTQ is unsafe. This accounts for the optimizations described in paragraphs %#1,2,3 of my memo on SHRKFY.)

```
                                                  (SETQQ OP3FLG OFFANGLES))
                                                 (T (SETQQ OP3FLG OPANGLE]
                                    (T
```

(* CDRLOOKAT is nil, so CARLOOKAT is the last argument of the form we are emulating.
It may also be the first, which we can detect if FIRSTARGFLG is T, in which case we are emulating a singleton.)

```
                                       (SELECTQ WORKFLAG
                                            (APPEND (SELECTQ OP2FLG
                                                         ((CONS NCONC NCONC1 LIST /NCONC /NCONC1)
                                                              (COND
                                                                 (FIRSTARGFLG
```

(* Since CDRLOOKAT is nil and FIRSTARGFLG is T, LOOKAT is a singleton and we are emulating an APPEND singleton. So brackets are not removed.)

```
                                                                        (SETQQ OP3FLG OPANGLE))
                                                                   ((OR (EQ OP2FLG 'LIST)
                                                                        (EQ OP2FLG 'CONS))
```

(* Otherwise, if CARLOOKAT is a LIST or CONS form, brackets can be removed, according to paragraph %#4 of the memo on SHRKFY optimizations.)

```
                                                                    (SETQQ OP3FLG OFFANGLES))
                                                                   (T
                                                     (* Otherwise brackets stay in.)
                                                                      (SETQQ OP3FLG OPANGLE))))
                                                 (APPEND
```

(* This branch accounts for the optimzations described in paragraph %#5 of the memo on SHRKFY.)

```
                                                                   (COND
                                                                      ((OR FIRSTARGFLG APPSINGFLG)
```

(* If CARLOOKAT is an append singleton then brackets are not removed, because it is the last argument of the APPEND form we are emulating. Or if FIRSTARGFLG is T, then since CDRLOOKAT is nil, we must be inside an append singleton, of which CARLOOKAT is the only argument, so brackets are not removed.)

```
                                                                        (SETQQ OP3FLG OPANGLE))
                                                                       (T (SETQQ OP3FLG OFFANGLES))))
                                                      NIL))
                                          (CONS          (* See paragraph %#6 of the memo on SHRKFY.)
                                                (SELECTQ OP2FLG
```

```
                                                    ((NCONC NCONC1 /NCONC /NCONC1)
                                                        (SETQQ OP3FLG OPANGLE))
                                                    ((CONS LIST)
                                                        (SETQQ OP3FLG OFFANGLES))
                                                    (APPEND (COND
                                                              (APPSINGFLG (SETQQ OP3FLG OPANGLE))
                                                              (T (SETQQ OP3FLG OFFANGLES))))
                                                   NIL))
                                          ((NCONC /NCONC)
                                                    (* See paragraph %#7 of the memo on SHRKFY.)
                                              (SELECTQ OP2FLG
                                                   ((LIST APPEND CONS)
                                                       (COND
                                                          (FIRSTARGFLG
                                                       (* We're emulating an NCONC singleton.)
                                                                 (SETQQ OP3FLG OPANGLE))
                                                          (CDDARLOOKAT
                                                       (* These cases all dwimify back correctly.)
                                                                 (SETQQ OP3FLG OFFANGLES))
                                                             (T (SETQQ OP3FLG OPANGLE))))
                                                   ((NCONC NCONC1 /NCONC /NCONC1)
                                                       (SETQQ OP3FLG OPANGLE))
                                                  NIL))
                                          ((NCONC1 /NCONC1)
```

(* There's no need to concern ourselves about bracket removal here.
Since CDRLOOKAT is NIL, and OPFLG = %'! or %'!!, and WORKFLAG = %'NCONC1 or %'/NCONC1, OPFLG must eq %'!!
(and FIRSTARGFLG must eq T, but we don't need to check for it)%, because NCONC1 never setq's OPFLG to %'!.)

```
                                                (SETQQ OP3FLG OPANGLE))
                                               NIL))))
                                  (LISTIT                        (* Brackets can't be removed.)
                                          [COND
                                             ([AND (LITATOM (CAR RESULTQ))
                                                   (FNTYP (CAR RESULTQ))
                                                   (NOT (BOUNDP (CAR RESULTQ]
```

(* something of the form <FOO |...| where FOO is name of functio and not the name f a variable wouldhave parens stuck back in it by dwimify.)

```
                                             (NCONC1 RESULTP (CAR LOOKAT)))
                                            (T (NCONC RESULTP (LIST '<)
                                                      RESULTQ
                                                      (LIST '>])
                                  NIL))
                        (T
```

(* RESULTQ has been pseudo-evaluated to NIL, so it disappears from or remains in RESULTP according to the rules desccribed in paragraphs %#8 through %#13 of the memo on SHRKFY.)

```
                (SELECTQ OPFLG
                     ((!! !)
                          (SELECTQ OP2FLG
                               ((APPEND NCONC LIST /NCONC)
                                    [COND
                                       ((AND FIRSTARGFLG (NULL (CDR CDRLOOKAT))
                                             CDRLOOKAT)
                                        (NCONC RESULTP (LIST OPFLG NIL])
                                ((CONS NCONC1 /NCONC1)
                                     (NCONC RESULTP (LIST OPFLG '< NIL '>)))
                               NIL))
                     (LISTIT (SELECTQ OP2FLG
                                  ((APPEND NCONC LIST /NCONC)
                                      (NCONC1 RESULTP NIL))
                                  ((CONS NCONC1 /NCONC1)
                                      (NCONC RESULTP (LIST '< NIL '>)))
                                 NIL))
                     NIL))))
        NIL)
A5
```

(* Here we add RESULTQ to RESULTP, according to the decision made in the previous major selectq, at A4.)

```
(SELECTQ OP3FLG
     (OFFANGLES (NCONC RESULTP RESULTQ))
     (OPANGLE [COND
                 [[AND (LITATOM (CAR RESULTQ))
                       (FNTYP (CAR RESULTQ))
                       (NOT (BOUNDP (CAR RESULTQ]
                  (NCONC RESULTP (LIST OPFLG (CAR LOOKAT]
                 (T (NCONC RESULTP (LIST OPFLG '<)
                           RESULTQ
                           (LIST '>])
     (CLISPIFY2CIT (SELECTQ OPFLG
                          ((! !!)
                               (NCONC RESULTP (LIST OPFLG (CLISPIFY2C CARLOOKAT))))
                          (LISTIT (NCONC1 RESULTP (CLISPIFY2C CARLOOKAT)))
                          NIL))
```

```
            NIL)
         [COND
            (RESULTR
```

(* RESULTR holds SHRKFY's translation of CDRLOOKAT, and of course does not have "implicit angle brackets" around it, so we just add it on to RESULTP.)

```
                   (NCONC RESULTP RESULTR))
            ((AND FIRSTARGFLG (EQ WORKFLAG 'NCONC1))
```

(* In this branch, since RESULTR is nil, it has either been pseudo-evaluated to nil or else we've been emulating an NCONC1 singleton. This branch makes sure (NCONC1 A) goes to (<!! A NIL>)%.)

```
             (NCONC1 RESULTP NIL))
            ((AND (EQ WORKFLAG 'APPEND)
                  CDRLOOKAT)
```

(* This branch makes sure that CARLOOKAT is copied. Since CDRLOOKAT is non nil, but RESULTR is nil, we know that RESULTR has been psuedo-evaluated to nil. If (APPEND A B (CONS) NIL (NCONC)) simply went to (<! A ! B >) and dwimified back to (APPEND A B)%, B would no longer be copied. So, for this case alone, we need to add a nil; the same problem does not arise within a CONS, LIST, or NCONC form. In fact, not doing anything in these cases allows us to optimize (CONS A (APPEND)) to (LIST A)%, and (NCONC A B (NCONC)) to (NCONC A B)%. On the other hand, (LIST A B (NCONC)) naturally goes to (<A B NIL>) and back to (LIST A B NIL)%.)

```
             (NCONC RESULTP (LIST '! NIL]
         (RETURN RESULTP])
```

(**SHRKFY2**

```
  [LAMBDA (LOOKAT WORKFLAG STAGEFLAG)
    (PROG (RESULTP)
          (SETQ RESULTP (LIST 'TEMPATOM))
          (SHRKFY LOOKAT WORKFLAG STAGEFLAG)
          (RETURN RESULTP])
```

(**WHILEDOUNTIL**

```
  [LAMBDA ($FORM)                                          (* DD%: "24-FEB-83 18:19")
    (PROG (PL FX FX1 CONDX TGO TEM WHILE DO UNTIL)
```

(* All syntatical patterns of the following format%: LABEL (COND (p1 e1...e2 (GO LABEL)) clause1...clause2) will be converted to the form%: LABEL (WHILE p1 DO e1...e2 (COND clause1...clause2))%. In addition, all patterns%: LABEL e1...e2 (COND (p1 (GO LABEL)) clause1...clause2) will be converted to the form%: LABEL (DO e1...e2 UNTIL (NOT p1)) (COND clause1...clause2)%. This function is invoked by CLISPIFY2B during CLISPIFY processing of a PROG.)

```
          (SETQ FX $FORM)
      TOP (COND
             ((NULL FX)
              (RETURN $FORM))
             ((NOT (ATOM (CAR FX)))
              (SETQ FX (CDR FX))
              (GO TOP)))
```

(* At this point a prog label has been detected and CADR of FX is a list. A test will now be made to determine if is an appropriate COND expression)

```
          (SETQ PL (CAR FX))
          (SETQ FX1 FX)
          (COND
             [(AND [LISTP (CAR (SETQ FX (CDR FX]
                   (EQ (CAR (SETQ CONDX (CAR FX)))
                       'COND)
                   (EQ [CAR (LISTP (CAR (SETQ TGO (LAST (CADR CONDX]
                       'GO)
                   (EQ (CADAR TGO)
                       PL)
                   (NULL (EDITFINDP (CADR CONDX)
                                    'RETURN T)))
              (SETQ DO (LDIFF (CDADR CONDX)
                              TGO))                        (* If the COND clause contains a predicate only, the DO expresstion will be omitted.)
              [SETQ WHILE (CONS 'WHILE (CONS (CAADR CONDX)
                                             (AND DO (CONS 'DO DO]
```

(* If the COND expression contains only one clause, the COND expression, constructed for the remaining clauses, is omitted.)

```
             (SETQ TEM (CONS WHILE (WHILEDO1 (CDDR CONDX]
             (T (GO TOP)))
          (RETURN (NCONC (LDIFF $FORM FX1)
                         (NCONC (CONS PL TEM)
                                (WHILEDOUNTIL (CDR FX])
```

(**WHILEDO1**

```
  [LAMBDA (X)
    (COND
       ((NULL X)
        NIL)
       ((AND (NULL (CDR X))
             (EQ (CAAR X)
                 T))
        (APPEND (CDAR X)))
       (T (LIST (CONS 'COND X])
```

(**CLDISABLE**

```
  [LAMBDA (OP)                                                    (* wt%: "14-NOV-78 01:44")
    (PROG (TEM FLG OP1 BRACKET)
          (SETQ OP1 (L-CASE OP))
          (SETQ BRACKET (GETP OP 'CLISPBRACKET))
          [COND
             ([AND (SETQ TEM (SELECTQ OP
                                  ((< ! >)                        (* I.S.OPR for JOIN uses <)
                                     'join)
                                  (+ 'sum)
                                  NIL))
                   (SETQ TEM (GETPROP TEM 'I.S.OPR]

          (* purpose of this is to convert the indicated i.s.opr to a lisp form instead of using infix notation before disabling the oprator,
          e.g. for SUM, I.S.OPR is ($$VAL_$$VAL+BODY) want to convert this to use IPLUS now)

              (RESETVARS (NOFIXFNSLST0 NOFIXVARSLST0)
                         (DWIMIFY0 (CAR TEM)
                                NIL
                                '(BODY $$VAL]
          [MAPC '(CLISPTYPE UNARYOP CLISPCLASS CLISPCLASSDEF CLISPNEG CLISPINFIX BROADSCOPE CLISPFORM I.S.OPR
                        CLISPWORD CLMAPS SETFN CLISPBRACKET)
                (FUNCTION (LAMBDA (X)

          (* does not remove LISPFN property, because this will be needed for explicit calls to CLISPLOOKUP from dwimify, e.g.
          for translating iterative statements using FROM and UNTIL, need to look up + and LT)

                             (COND
                                ((/REMPROP OP X)
                                 (SETQ FLG T)))
                             (COND
                                ((/REMPROP OP1 X)
                                 (SETQ FLG T]
          [MAPC '(I.S.OPRLST CLISPFORWORDSPLST CLISPINFIXSPLST)
                (FUNCTION (LAMBDA (X)
                             (/SETATOMVAL X (REMOVE OP (GETATOMVAL X]
          [COND
             ((MEMB OP CLISPCHARS)
              (/SETATOMVAL 'CLISPCHARS (REMOVE OP CLISPCHARS))
              (/SETATOMVAL 'CLISPCHARRAY (MAKEBITTABLE CLISPCHARS))
              (SETQ FLG T)
              (SELECTQ OP
                   (- (CLDISABLE '+-))
                   (+- (CLDISABLE '-))
                   (! (CLDISABLE '!!))
                   NIL
                   (COND
                      (BRACKET (CLDISABLE (CAR BRACKET))
                               (CLDISABLE (CADR BRACKET))
                               (AND (SETQ TEM (LISTGET1 BRACKET 'SEPARATOR))
                                    (CLDISABLE TEM]
          (RETURN (AND FLG OP])
```

)

(RPAQ? **FUNNYATOMLST**)

(RPAQ? **CLREMPARSFLG**)

(RPAQ? **CL%:FLG** T)

(RPAQ? **CLISPIFYPACKFLG**)

(RPAQ? **CLISPIFYENGLSHFLG**)

(RPAQ? **CLISPIFYUSERFN**)

(RPAQQ **CAR/CDRSTRING** "CAR/21-")

```
(ADDTOVAR EDITMACROS
          (CL NIL (BIND (IF (NULL (CDR L))
                            [(IF (MEMB (%## 1)
                                       LAMBDASPLST)
```

```
                                ((MARK %#3)
                                 3 UP)
                                ((E (PROGN (SETQQ COM CL)
                                           (PRINT 'can't T T)
                                           (ERROR!]
                            NIL)
                        [IF (TAILP (SETQ %#1 (%##))
                                   (%## !0 (E (SETQ %#2 L)
                                              T)))
                            ((I %: (CLISPIFY %#1 %#2))
                             (LO 1))
                            ((COMS (CONS '%: (CLISPIFY %#1 %#2))
                                   (AND (LISTP (%## 1))
                                        1]
                        (IF %#3 ((\ %#3))
                            NIL))))
```

(ADDTOVAR **EDITCOMSA** CL)

(PUTPROPS **ADD1 CLISPFORM** (IPLUS * 1))

(PUTPROPS **SUB1 CLISPFORM** (IPLUS * -1))

(PUTPROPS **NEQ CLISPFORM** (NOT (EQ . *)))

(PUTPROPS **CONS CLISPBRACKET** <)

(PUTPROPS **LIST CLISPBRACKET** <)

(PUTPROPS **APPEND CLISPBRACKET** <)

(PUTPROPS **NCONC CLISPBRACKET** <)

(PUTPROPS **NCONC1 CLISPBRACKET** <)

(PUTPROPS **/NCONC CLISPBRACKET** <)

(PUTPROPS **/NCONC1 CLISPBRACKET** <)

(PUTPROPS **~EQUAL CLISPTYPE** NIL)

(PUTPROPS **~MEMBER CLISPTYPE** NIL)

(PUTPROPS **~MEMB CLISPTYPE** NIL)

(PUTPROPS **MAPC CLMAPS** (in . do))

(PUTPROPS **MAP CLMAPS** (on . do))

(PUTPROPS **MAPCAR CLMAPS** (in . collect))

(PUTPROPS **MAPLIST CLMAPS** (on . collect))

(PUTPROPS **MAPCONC CLMAPS** (in . join))

(PUTPROPS **MAPCON CLMAPS** (on . join))

(PUTPROPS **SUBSET CLMAPS** (in . subset))

```
(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY

(BLOCK%: CLISPIFYBLOCK CLISPIFYFNS CLISPIFY CLISPIFY1 CLISPIFY2 CLISPIFY2A CLISPIFY2B CLISPIFY2C CLISPIFY2D
       CLISP3 CLISP3A CLISP3B CLISPACKUP CLISP3C CLISP4 CLISPCOND CLISPCOND1 CLISPAND CLISPAND1 CLISPIFYNOT
       CLISPIFYMATCHUP CLREMPARS CLISPIFYCROPS0 CLISPIFYCROPS CLISPIFYCROPS1 CLISPIFYRPLAC CLISPIFYMAPS CLMAPS1
       CLMAPS2 SHRIEKIFY SHRKFY SHRKFY2 CLISPIFYLOOKUP CLSTOPSCAN? WHILEDOUNTIL WHILEDO1
       (ENTRIES CLISPIFYFNS CLISPIFY CLISPACKUP CLISPIFYMATCHUP CLISPIFY2A CLISP3A)
       (SPECVARS EXPR VARS DWIMIFYFLG DWIMIFYING DWIMIFY0CHANGE)
       (LOCALFREEVARS DECLST CLTYP0 OPR0 LST SEG TAIL FORM PARENT SUBPARENT NOVALFLG NEGFLG RESULTP SAFEFLAG
              VARS CLISPISTATE TYPE-IN? SIDES CLISPIFYFN)
       (GLOBALVARS CAR/CDRSTRING CL%:FLG CLISPARRAY CLISPCHARRAY CLISPCHARS CLISPFLG CLISPIFYENGLSHFLG
              CLISPIFYPACKFLG CLISPIFYSTATS CLISPIFYUSERFN CLISPISNOISEWORDS CLISPISVERBS CLISPTRANFLG
              CLREMPARSFLG COMMENTFLG DWIMFLG FILELST FUNNYATOMLST GLOBALVARS LCASEFLG)
       (RETFNS CLISPIFY2B)
       (NOLINKFNS CLISPIFYUSERFN))

(BLOCK%: NIL LOWERCASE (GLOBALVARS CHCONLST LCASEFLG))

(BLOCK%: NIL CLDISABLE (GLOBALVARS CLISPCHARS CLISPCHARRAY NOFIXFNSLST0 NOFIXVARSLST0))

(BLOCK%: NIL (GLOBALVARS CLISPISNOISEWORDS CLISPISVERBS CLISPISWORDSPLST))
)
```

(**LOWERCASE** T)

```
(DECLARE%: DOEVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE
```

```
(RECORD MATCHUP ((NIL . SUBJ)
                 (NIL . OBJ)))
)
)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA CLISPIFYFNS)

(ADDTOVAR NLAML )

(ADDTOVAR LAMA )
)

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

FUNCTION INDEX

PROPERTY INDEX

VARIABLE INDEX

RECORD INDEX