N E W S L E T T E R # 2

March 1980

This is the second edition of The Soft Warehouse Newsletter. It is devoted to bringing our customers up to date on the latest news, updates, and products of The Soft Warehouse. In addition, it provides a medium for the growing community of **muLISP** and **muSIMP/muMATH** users to exchange ideas and general purpose utility programs. We welcome short contributions, or announcements regarding applications for our products.

Since our November Newsletter was issued, we have received some valuable suggestions from customers concerning corrections and improvements to both our reference manuals. Also an excellent muSIMP function definition pretty printer was contributed and is included in this letter. First, however, here are some **hot** news items:

MICROSOFT to Distribute muLISP and muMATH

The Soft Warehouse has licensed to **Microsoft** of Bellevue, Washington the rights to distribute both our current products. As you may have noticed from their ads in the March and April issues of BYTE, they have begun a major sales effort. In addition to the CP/M version, Microsoft will soon be distributing muMATH for the TRS-80, thus making symbolic math available to a much larger audience. Costing much less, the TRS-80 version will not have all the features of its CP/M counter-part. We will continue to distribute the CP/M version of our systems until next fall, at which time Microsoft will gain exclusivity.

Microsoft is widely recognized, especially with OEMs, for high quality software and of course they have a much more extensive marketing ability than The Soft Warehouse. This should greatly accelerate the recognition and acceptance of LISP and computer algebra among the computing public. Most important from our standpoint and yours, the arrangement frees us to concentrate on what we do best -- innovative software development for small computers.

1980 LISP Conference

In case you haven't heard, there will be a LISP conference on the Stanford University campus from August 24 to 27. The program committee includes many of the important names in the field of applicative languages including the man who started it all, John McCarthy. The topics include applicative and object oriented languages, LISP program verification, the design and implementation of hardware LISP machines, and computer algebra. Information is available from

Conference Head:
John R. Allen
Stanford Artificial Intelligence Lab
Stanford University
Stanford, CA 94305
(415) 497-4971

Local Arrangements:
Dr. Ruth E. Davis
Dept. of EECS
University of Santa Clara
Santa Clara, CA 95053
(408) 984-4358

The Small Systems Group's Software Survey

The Small Systems Group is making an extensive survey of CP/M based software systems. As both authors and users of such software, we strongly support this project. Consequently, we have enclosed a copy of their **Grass Roots Software Evaluation Survey** form. If you wish to participate, just fill it out and send it directly to The Small Systems Group, Box 5429, Santa Monica, CA 90405.

* * * * * * * * * * * * **The muMATHematician** * * * * * * * * * * * *

Errors in the Matrix Package

Two bugs were found in the 12/27/79 and 7/16/79 version of the muMATH-79 source file MATRIX.ARR. If you have either of these versions, the errors can be corrected using your systems text editor as follows:

1. The 5th line of the function INPROD (i.e. the line numbered 335) should be changed to read:

```
WHEN EMPTY (LEX2),  EX1 EXIT
```

2. The other bug only occurs when RAGGED is FALSE during matrix division. In order to locally set RAGGED to TRUE in \ and IDMAT, make the following two changes:

```
FUNCTION IDMAT (EX1, EX2,
  RAGGED),
    RAGGED: TRUE,
    EX2: LIST (1),
      . . .
```

```
FUNCTION \ (EX1, EX2, RAGGED, EX3
  EX4, LEX1, LEX2, LEX3, LEX4),
  RAGGED: TRUE,
  WHEN ARRAY (EX1)
    . . .
```

The DEAR ALGy Column

DEAR ALGY: I really like your recursive and ragged arrays, which make those of other languages seem square. However, I have one assigning request to make: Please provide a means of changing the value of an individual element in an array. -- Harray

DEAR HARRAY: Here is a solution which also provides the convenience of initializing unassigned elements to zero:

```
PROPERTY INFIX, :, COND (
  WHEN NAME (EX1), LIST (':, EX1, PARSE (SCAN,20)) EXIT,
  WHEN FIRST(EX1) = 'SUBSCR,
    LIST (UPDATE, SECOND(EX1), RREST(EX1), PARSE(SCAN,20)) EXIT,
  WHEN SYNTAX () EXIT) $

SUBROUTINE UPDATE (EX1, LEX1, EX2),
  ASSIGN (EX1, UPDATE1 (EVAL(EX1), LEX1)),
ENDSUB $

FUNCTION UPDATE1 (EX3, LEX1),
  WHEN ATOM (LEX1), EVAL (EX2) EXIT,
  WHEN ARRAY (EX3)  AND  POSITIVE (FIRST(LEX1)),
    ADJOIN (FIRST(EX3), UPDATE2(REST(EX3),FIRST(LEX1))) EXIT,
  ? (LIST ('_, EX1, EX2)),
ENDFUN $

FUNCTION UPDATE2 (LEX2, EX4),
  BLOCK
    WHEN ATOM (LEX2), LEX2: LIST(0) EXIT,
  ENDBLOCK,
  WHEN EX4 = 1,
    ADJOIN (UPDATE1(FIRST(LEX2),REST(LEX1)), REST(LEX2)) EXIT,
  ADJOIN (FIRST(LEX2), UPDATE2 (REST(LEX2), EX4-1)),
ENDFUN $
```

DEAR ALGY: I tried making "=" work for both equations and predicates as suggested in Newsletter 1, but it is not totally satisfactory. -- Egalitarian.

DEAR EGALITARIAN: Yes, it was greedy of me to attempt reading the user's mind to determine what kind of "=" is intended. We now recommend the use of == without the double quotes to designate equations. To do this change all occurrences of " = " to "==" in the files EQN.ALG, SOLVE.EQN, and ARRAY.ARI; change the date in all three files to 03/31/80; and make the following addition to EQN.ALG:

```
PROPERTY INFIX, =, EQPARSE (SCAN) $

FUNCTION EQPARSE (EX2),
  WHEN EQ (EX2, '=), LIST ('"==", EX1, PARSE (SCAN(), 80)) EXIT,
  LIST ('=, EX1, PARSE (SCAN, 80))
ENDFUN $
```

A Summation Package

The source file SIGMA.ALG computes the closed-form representation for the sums and products of series. Given an expression EXPN, containing an indeterminate J, the summation of EXPN from J equal M to N is given by: SIGMA (EXPN, J, M, N). For example, the closed-form representation for

$$\sum_{j=1}^{n} (cj^2 + 2^j)$$

in terms of c and f, courtesy of file SIGMA.ALG, is:

$$cn/6 + cn^2/2 + cn^3/3 + 2^{(1+n)} - 2$$

```
% File:  SIGMA.ALG  (c)       03/01/82    The Soft Warehouse %


FUNCTION LINCF (EX1, INDET),
  EX1: (EX1 - EVSUB (EX1, INDET, 0)) / INDET,
  WHEN FREE (EX1, INDET), EX1 EXIT
ENDFUN$

FUNCTION FREE (EX1, EX2),
  WHEN EX1 = EX2, FALSE EXIT,
  WHEN ATOM (EX1) EXIT,
  LOOP
    WHEN NOT FREE (FIRST (EX1), EX2), FALSE EXIT,
    EX1: REST (EX1),
    WHEN ATOM (EX1) EXIT,
  ENDLOOP,
ENDFUN$

FUNCTION SIGMA (EX1, INDET, EX2, EX3,
  % Local: % EX4, NUMNUM, DENDEN, DENNUM, NUMDEN, PWREXPD, LOGEXPD),
  WHEN INTEGER(EX2) AND INTEGER(EX3),
    EX4: 0,
    LOOP
      WHEN EX2 > EX3, EX4 EXIT,
      EX4: EX4 + EVSUB(EX1,INDET,EX2),
      EX2: EX2 + 1
    ENDLOOP EXIT,
  NUMNUM: DENDEN: 30, DENNUM: -30, NUMDEN: 0, PWREXPD: LOGEXPD: 6,
  WHEN FREE (EX4:ANTIDF(EX1), ANTIDF),
    BLOCK
      WHEN INTEGER(#LIM) AND EX2 EQ MINF,
      EX1: LIM (EX4, INDET, EX2) EXIT,
      EX1: EVSUB (EX4, INDET, EX2) ENDBLOCK,
    BLOCK
      WHEN INTEGER(#LIM) AND EX3 EQ PINF, EX4: LIM(EX4,INDET,EX3) EXIT,
      EX4: EVSUB (EX4, INDET, EX3+1) ENDBLOCK,
    WHEN INTEGER(#LIM) AND (EX2 EQ MINF OR EX3 EQ PINF),
```

```
      LIM (EX4-EX1) EXIT,
    EX4-EX1 EXIT,
  WHEN APPLY(GET('SIGMA,FIRST(EX1)), ARGEX(EX1)) EXIT,
  LIST('SIGMA, EX1,INDET,EX2,EX3),
ENDFUN$

FUNCTION ANTIDF (EX1),
  WHEN EX1 = INDET, EX1*(-1+EX1)/2 EXIT,
  WHEN FREE(EX1,INDET), INDET*EX1 EXIT,
  SIMPU(ANTIDF,EX1)
ENDFUN$

PROPERTY ANTIDF, +, FUNCTION (EX1, EX2),
  WHEN ZERO (EVSUB(EX1,INDET,INDET+1) + EX2), -EX1  EXIT,
  WHEN ZERO (EVSUB(EX2,INDET,INDET+1) + EX1), -EX2  EXIT,
  ANTIDF(EX1) + ANTIDF(EX2),
ENDFUN$

PROPERTY ANTIDF, *, FUNCTION (EX1, EX2),
  WHEN FREE(EX1,INDET), EX1*ANTIDF(EX2)  EXIT,
  WHEN FREE(EX2,INDET), EX2*ANTIDF(EX1)  EXIT,
ENDFUN$

PROPERTY ANTIDF, ^, FUNCTION (EX1, EX2, EX3),
  WHEN EX1 = INDET AND POSITIVE(EX2),
    (EX3:PROD(INDET-'##,'##,0,EX2-1)) * (INDET-EX2)/(1+EX2)
      + ANTIDF(EX1^EX2-EX3)  EXIT,
  WHEN FREE(EX1,INDET) AND (EX3: LINCF(EX2,INDET)),
    EX1^EX2/(EX1^EX3 - 1) EXIT,
ENDFUN$

FUNCTION PROD (EX1, INDET, EX2, EX3,
    % Local: %  EX4, LOGEXPD),
  WHEN INTEGER(EX2) AND INTEGER(EX3),
    EX4: 1,
    LOOP
      WHEN EX2 > EX3, EX4 EXIT,
      EX4: EX4 * EVSUB(EX1,INDET,EX2),
      EX2: EX2 + 1,
    ENDLOOP EXIT,
  WHEN FREE (EX4:ANTIDV(EX1), ANTIDV),
    WHEN INTEGER(#LIM), LIM(EX4,INDET,EX3+1) / LIM(EX4,INDET,EX2) EXIT,
    EVSUB(EX4,INDET,EX3+1) / EVSUB(EX4,INDET,EX2) EXIT,
  WHEN APPLY(GET('PROD,FIRST(EX1)), ARGEX(EX1)) EXIT,
  LIST ('PROD, EX1, INDET, EX2, EX3)
ENDFUN$

RDS()$
```

A muMATH Function Deparser and Pretty Printer

The attached muMATH function definition pretty printer was contri-buted by T. B. Robinson of Research Machines Ltd., Oxford, England. The basic user level functions are DISPFUN and WRITEFILE. DISPFUN outputs a formatted definition of its argument on the console. WRITEFILE ([fname,ftype,drive], fun1, fun2, ...) not only writes out the definition of **fun$_i$** on the given file, but any values and/or properties associated with fun$_i$.

```
% File:  UNPARSE.MUS          02/12/82          The Soft Warehouse %


% This muSIMP function unparser and pretty-print package was written
by T. B. Robinson of Research Machines Ltd., Oxford, England. %

FUNCTION DISPFUN (F#U#N#, LEX1),
   NEWLINE (2),
   UNPARSE (0, FALSE, LIST (GETD(F#U#N#))),
   PRINT (""),
ENDFUN$

F#U#N#: ""$

FUNCTION PRTARGS (INDENT, LEX1),
   SPACES (1),
   WHEN EMPTY (LEX1), PRINT ("()") EXIT,
   WHEN ATOM (LEX1),
     QUOTEPRINT (LEX1) EXIT,
   WHEN DPAIR (LEX1),
     PRINTDPAIR (LEX1) EXIT,
   PRINT (LPAR),
   LOOP
     UNPARSE (INDENT, FALSE, LIST (FIRST (LEX1))),
     POP (LEX1),
     WHEN EMPTY (LEX1) EXIT,
     PRINT (COMMA),
     SPACES (1),
   ENDLOOP,
   PRINT (RPAR),
ENDFUN$

PROPERTY UNPARSE, ', FUNCTION (INDENT, LEX1),
   PRINT (''),
   PRTARGS (INDENT, FIRST(LEX1)),
   TRUE,
ENDFUN$

PROPERTY UNPARSE, NOT, FUNCTION (INDENT, LEX1),
   PRINT ('NOT),
   SPACES (1),
   UNPARSE (INDENT, FALSE, LEX1),
   TRUE,
ENDFUN$
```

```
PROPERTY UNPARSE, FUNCTION, FUNCTION (INDENT, LEX1),
   PRINT ('FUNCTION),
   SPACES (1),
   PRINT (F#U#N#),
   F#U#N#: "",
   PRTARGS (INDENT, FIRST (LEX1)),
   UNPARSE (INDENT+2, TRUE, REST(LEX1)),
   PRNTLINE (INDENT),
   PRINT ('ENDFUN),
ENDFUN$

PROPERTY UNPARSE, SUBROUTINE, FUNCTION (INDENT, LEX1),
   PRINT ('SUBROUTINE),
   SPACES (1),
   PRINT (F#U#N#),
   F#U#N#: "",
   PRTARGS (INDENT, FIRST(LEX1)),
   UNPARSE (INDENT+2, TRUE, REST(LEX1)),
   PRNTLINE (INDENT),
   PRINT ('ENDSUB),
ENDFUN$

PROPERTY UNPARSE, LOOP, FUNCTION (INDENT, LEX1),
   PRINTLINE ('LOOP),
   SPACES (INDENT+2),
   UNPARSE (INDENT+2, FALSE, LEX1),
   PRNTLINE (INDENT),
   PRINT ('ENDLOOP)
ENDFUN$

FUNCTION UNPARSE (INDENT, EOL, LEX1, LEX2),
   WHEN EMPTY (LEX1) EXIT,
   LEX2: FIRST (LEX1),
   WHEN DPAIR (LEX2),
     PRINTDPAIR (LEX2) EXIT,
   WHEN ATOM (LEX2),
     BLOCK
       WHEN EOL,
       PRNTLINE (INDENT) EXIT,
     ENDBLOCK,
     QUOTEPRINT (LEX2) EXIT,
   BLOCK
     WHEN EOL, PRNTLINE (INDENT) EXIT,
   ENDBLOCK,
   WHEN ATOM (FIRST (LEX2)),
     UNPARSEFUN (INDENT, LEX2),
     UNPARSE (INDENT, TRUE, REST(LEX1)) EXIT,
   WHEN ATOM (FIRST (FIRST (LEX2))),
     UNPARSEWHEN (INDENT, LEX2),
     UNPARSE (INDENT, TRUE, REST(LEX1)) EXIT,
   UNPARSEBLOCK (INDENT, LEX2),
   UNPARSE (INDENT, TRUE, REST(LEX1)),
ENDFUN$
```

```
FUNCTION UNPARSEFUN (INDENT, LEX1, LEX2),
   LEX2: FIRST (LEX1),
   WHEN INTEGER (LEX2),
     PRINT (''),
     PRTARGS (INDENT, LEX1) EXIT,
   WHEN APPLY (GET(UNPARSE,LEX2), LIST(INDENT,REST(LEX1))) EXIT,
   WHEN GET ('LBP, LEX2),
     WHEN EMPTY (RREST(LEX1)),
       PRINT (LEX2),
       SPACES (1),
       UNPARSE (INDENT, FALSE, REST(LEX1)) EXIT,
     UNPARSE (INDENT, FALSE, LIST(SECOND(LEX1))),
     SPACES (1),
     PRINT (LEX2),
     SPACES (1),
     UNPARSE (INDENT, FALSE, RREST(LEX1)) EXIT,
   PRINT (LEX2),
   PRTARGS (INDENT, REST(LEX1)),
ENDFUN$

FUNCTION UNPARSEWHEN (INDENT, LEX1),
   PRINT ('WHEN),
   SPACES (1),
   UNPARSE (INDENT+2, FALSE, LEX1),
   SPACES (1),
   PRINT ('EXIT),
ENDFUN$

FUNCTION UNPARSEBLOCK (INDENT, LEX1),
   PRINTLINE ('BLOCK),
   SPACES (INDENT+2),
   UNPARSE (INDENT+2, FALSE, LEX1),
   PRNTLINE (INDENT),
   PRINT ('ENDBLOCK),
ENDFUN$

FUNCTION DPAIR (LEX1),
   WHEN ATOM (LEX1), FALSE EXIT,
   WHEN EMPTY (REST (LEX1)), FALSE EXIT,
   ATOM (FIRST (LEX1)) AND ATOM (REST (LEX1)),
ENDFUN$

FUNCTION PRINTDPAIR (LEX1),
   PRINT (LPAR),
   QUOTEPRINT (FIRST (LEX1)),
   PRINT (". "),
   QUOTEPRINT (REST (LEX1)),
   PRINT (RPAR),
ENDFUN$

FUNCTION PRNTLINE (INDENT),
   PRINTLINE (COMMA),
   SPACES (INDENT),
ENDFUN$
```

```
MOVD (PRINT,QUOTEPRINT)$

% The following is an optional print package for names which
contain separator and/or break characters. Such names must be
quoted in order to read then back in. This is essential if the
output is to be sent to a file and subsequently read back using
the RDS command. %

FUNCTION QUOTEPRINT (EX1, EX2, LEX1, LEX2),
   WHEN INTEGER (EX1),
     PRINT (EX1) EXIT,
   WHEN LENGTH(EX1) = 0,
     PRINT ("""""""") EXIT,
   LEX1: EXPLODE (EX1),
   BLOCK
     WHEN DIGIT (FIRST (LEX1)),
       EX2: TRUE EXIT,
     LEX2: LEX1,
     LOOP
       WHEN WILDCHAR (POP (LEX2)),
       EX2: TRUE EXIT,
       WHEN EMPTY (LEX2) EXIT,
     ENDLOOP,
   ENDBLOCK,
   WHEN NOT EX2, PRINT (EX1) EXIT,
   PRINT("""""),
   LOOP
     BLOCK
       WHEN PRINT (POP(LEX1)) = """",
       PRINT("""""") EXIT,
     ENDBLOCK,
     WHEN EMPTY (LEX1) EXIT,
   ENDLOOP,
   PRINT("""""),
ENDFUN$

FUNCTION WILDCHAR (EX1),
   MEMBER (EX1, LIST ('" ", '!, '""""", '$, '"%", '&, '', LPAR,
     RPAR, '*, '+, COMMA, '-, '., '/, ':, ';, '<, '=, '>, '?,
     '[, '\, '], '^, '{, '|, '}, '~ )),
ENDFUN$

FUNCTION DIGIT (EX1),
   MEMBER (EX1, '("0","1","2","3","4","5","6","7","8","9")),
ENDFUN$

%   Optional WRITEFILE to save FUNCTIONs and SUBROUTINEs on disk. %

PROPERTY PREFIX, [, MATCH (])$

SUBROUTINE WRITEFILE LEX1
   WRITEFILE1 (FIRST(LEX1), REST(LEX1)),
ENDSUB$

FUNCTION WRITEFILE1 (LEX1, LEX2),
```

```
   WRS (FIRST(LEX1), SECOND(LEX1), THIRD(LEX1)),
   LOOP
     WHEN EMPTY (LEX2) EXIT,
     PRINTFUN (FIRST (LEX2)),
     PRINTVAL (FIRST (LEX2)),
     PRINTPROPS (POP (LEX2)),
   ENDLOOP,
   NEWLINE (2),
   PRINT ("RDS ()$"),
   WRS (),
   FIRST (LEX1),
ENDFUN$

FUNCTION PRINTFUN (EX1),
   WHEN ATOM (GETD (EX1)) EXIT,
   DISPFUN (EX1),
   PRINT ('$),
ENDFUN$

FUNCTION PRINTVAL (EX1),
   WHEN FIRST (EX1) = EX1 EXIT,
   NEWLINE (2),
   QUOTEPRINT (EX1),
   PRINT (" : "),
   UNPARSE (2, FALSE, LIST(FIRST(EX1))),
   PRINTLINE ('$),
ENDFUN$

FUNCTION PRINTPROPS (EX1,
    % Locals % EX2, LEX1),
   LEX1: REST (EX1),          % LEX1: PROPERTY LIST OF EX1 %
   LOOP
     WHEN ATOM (LEX1) EXIT,
     EX2: POP (LEX1),         % EX2: A PROPERTY OF EX1 %
     NEWLINE (2),
     PRINT ('PROPERTY),
     SPACES (1),
     QUOTEPRINT (EX1),
     PRINT (", "),
     PRINT (FIRST(EX2)),
     BLOCK
       WHEN REST(EX2) EQ COMPRESS (LIST (EX1, FIRST(EX2))),
       UNPARSE (2, TRUE, LIST (GETD(REST(EX2)))) EXIT,
       UNPARSE (2, TRUE, LIST (REST(EX2))),
     ENDBLOCK,
     PRINTLINE ('$),
   ENDLOOP,
ENDFUN$

RDS ()$
```

A More Efficient DEPTH Function

Robert England of Mississauga, Ontario, Canada has pointed out that our UTILITY library definition of DEPTH is ridiculously inefficient. The definition of DEPTH given in the 8/6/79 version of UTILITY.LIB unnecessarily recomputes the depths of subtrees. This results in an exponential growth problem, as is clearly revealed by use of the TRACE facility. The following is an alternative definition which in addition to being efficient is an elegant example of the use of helper functions.

```
PUTD (MAX (LAMBDA (M N)
  ((GREATERP M N) M)
  N ))

PUTD (DEPTH (LAMBDA (EXPN)
  ((ATOM EXPN) 0)
  (PLUS 1 (MAX (DEPTH (CAR EXPN)) (DEPTH (CDR EXPN)))) ))
```

A Function Definition Pretty Print Package

As a service to those LISP users who are accustomed to interactively developing programs within LISP and then permanently saving the program on disks, the accompanying source listing for the file PRTDEF.LIB is attached.

Function PRETTYPRT prints a LAMBDA defined function in an easy to read format. It makes use of indentation and spacing of parentheses to enhance the underlying structure of the definition. Its first argument is a function name or list of names. Its second, third, and fourth give the desired output file Name, Type, and Drive. If these parameters are omitted, the output is directed to the console instead. If the function definition uses muLISP names which contain break or separator characters, such as blanks, commas, parentheses, etc., double quotes will have to be manually added later by using your system's text editor.

In addition PRTDEF.LIB illustrates several other useful features. The function MAPC is a general purpose mapping function which successively applies its second argument to the elements of its first argument. The value it returns is NIL.

Many LISP users prefer an EVAL type executive driver loop as distinct from an EVALQUOTE loop. For such people the CBN function DEFUN is a great convenience in defining functions. As listed in PRTDEF.LIB, DEFUN's first argument is the function name and the second is the function's definition. The definition is a list including the LAMBDA or NLAMBDA identifier, the formal argument list, and then the function body. The following definition of APPEND is an example of its use:

```
          (DEFUN APPEND (LAMBDA (LST1 LST2)
            ((NULL LST1) LST2)
            (CONS (CAR LST1) (APPEND (CDR LST1) LST2)) ))

%File:  PRTDEF.LIB           06/24/80    The Soft Warehouse %


(PUTD DEFUN (QUOTE (NLAMBDA (FUNC DEF)
  (PUTD FUNC DEF)
  FUNC )))

(DEFUN MAPC (LAMBDA (ALST FUNCT)
  (LOOP
    ((NULL ALST) NIL)
    (FUNCT (CAR ALST))
    (SETQ ALST (CDR ALST)) ) ))

(DEFUN PRETTYPRT (NLAMBDA (DEFS FNAME FTYPE FDRIVE)
  (WRS FNAME FTYPE FDRIVE)                % Open the file %
  ( ((AND (ATOM DEFS) (NOT (NULL DEFS)))
      (PRTDEF DEFS))
    (MAPC DEFS PRTDEF) )                  % Print the file %
  (TERPRI)
  (PRIN1 (QUOTE (RDS)))
  (WRS)                                   % Close the file %
  T ))

(DEFUN PRTDEF (LAMBDA (DEF
    LINELENGTH INDENT2 )
  (TERPRI)
  ((ATOM (GETD DEF))
    (PRIN2 DEF)
    (PRINT " is not a LAMBDA defined function.")
    "" )
  (SETQ LINELENGTH (LINELENGTH))
  ( ((PLUSP INDENT))
    ((LESSP LINELENGTH 60)
      (SETQ INDENT 1) )
    (SETQ INDENT 2) )
  (SETQ INDENT2 (TIMES INDENT 2))
  (PRIN1 "(DEFUN ")
  (PRIN2 DEF)
  (SETQ DEF (GETD DEF))
  (PRIN1 " (")
  (PRIN2 (CAR DEF))
  (SPACE-TAB 0)
  (PRTLST (CADR DEF) INDENT2)
  (PRTBDY (CDDR DEF) INDENT)
  (PRINT "))")
  "" ))

(DEFUN PRTBDY (LAMBDA (BDY TAB)
  ((NULL BDY))
```

```
  (TERPRI-TAB TAB)
  (PRTBDY1 BDY TAB) ))

(DEFUN PRTBDY1 (LAMBDA (BDY TAB)
  (LOOP
    (PRTTSK (CAR BDY) TAB)
    (SETQ BDY (CDR BDY))
    ((NULL BDY)
      (SPACE-TAB TAB) )
    (TERPRI-TAB TAB) ) ))

(DEFUN PRTTSK (LAMBDA (TSK TAB)
  ((ATOM TSK)
    (PRIN2 TSK) )
  ((ATOM (CAR TSK))
    ((MEMBER (CAR TSK) (QUOTE (LOOP COND PROGN PROG1)))
      (PRIN1 "(")
      (PRTATM (CAR TSK) (PLUS TAB INDENT2))
      (PRTBDY (CDR TSK) (PLUS TAB INDENT))
      (PRTATM ")" (PLUS TAB INDENT) T) )
    (PRTLST TSK (PLUS TAB INDENT2)) )
  ((ATOM (CAAR TSK))
    (PRIN1 "(")
    (PRTLST (CAR TSK) (PLUS TAB INDENT2))
    (PRTBDY (CDR TSK) (PLUS TAB INDENT))
    (PRTATM ")" (PLUS TAB INDENT) T) )
  (PRIN1 "( ")
  (PRTBDY1 TSK (PLUS TAB INDENT))
  (PRTATM ")" (PLUS TAB INDENT) T) ))

(DEFUN PRTLST (LAMBDA (LST TAB
    TSK NEWLINE )
  (PRIN1 "(")
  (LOOP
    ((ATOM LST)
      ((NOT NEWLINE)
      (PRTATM ")" TAB T) )
      (PRTATM " )" TAB T) )
    (SETQ TSK (CAR LST))
    (SETQ LST (CDR LST))
    ( ((FIT-PRT TSK)
      ((ATOM LST))
      (SPACE-TAB TAB) )
      (SETQ NEWLINE T)
      ((ATOM TSK)
      (PRTATM TSK TAB)
      ((ATOM LST))
      (SPACE-TAB TAB) )
      (TERPRI-TAB TAB)
      ((FIT-PRT TSK)
      ((ATOM LST))
      (SPACE-TAB TAB) )
      (PRTLST TSK (PLUS TAB INDENT2))
      ((ATOM LST))
      (TERPRI-TAB TAB) ) ) ))
```

```
(DEFUN PRTATM (LAMBDA (ATM TAB PRIN1)
  ( ((LESSP (PLUS (SPACES) (LENGTH ATM)) LINELENGTH))
    ((LESSP (PLUS TAB (LENGTH ATM)) LINELENGTH)
      (TERPRI-TAB TAB) ) )
  (PRIN1 ATM) ))

(DEFUN SPACE-TAB (LAMBDA (TAB)
  ((ZEROP (SPACES 1))
    (SETQ NEWLINE T)
    (TERPRI-TAB) ) ))

(DEFUN TERPRI-TAB (LAMBDA (TAB)
  (TERPRI)
  (SPACES TAB) ))

(DEFUN FIT-PRT (LAMBDA (LST)
  ((LESSP (PLUS (SPACES) (PRTLEN LST)) LINELENGTH)
    (PRIN2 LST)
    T ) ))

(DEFUN PRTLEN (LAMBDA (TSK PRIN1
    LEN)
  ((ATOM TSK)
    (LENGTH TSK) )
  (SETQ LEN 1)
  (LOOP
    (SETQ LEN (PLUS LEN (PLUS (PRTLEN (CAR TSK)) 1)))
    (SETQ TSK (CDR TSK))
    ((NULL TSK)
      LEN )
    ((ATOM TSK)
      (PLUS LEN (PLUS 3 (PRTLEN TSK))) ) ) ))

(DEFUN PRIN2 (LAMBDA (EXPN PRIN1)
  (PRIN1 EXPN) ))

(RDS)
```