N E W S L E T T E R # 9

October 1983

Aloha from Hawaii! The Soft Warehouse Newsletter provides you with information on new Soft Warehouse products, and software extensions or corrections to existing products. In addition, the newsletter is a medium for the exchange of ideas and application programs within the growing community of **muMATH** and **muLISP** users.

If you would like to subscribe, or extend your subscription to the Newsletter for three issues beyond the expiration number on your mailing label, please send $6 ($10 for orders from outside the U.S. or Canada) by check, VISA, or Master Card to The Soft Warehouse, P.O. Box 11174, Honolulu, Hawaii, 96828, U.S.A. A complete set of back issues is available on request for $15.

Announcing muMATH-83 and muLISP-83

Wondered why you haven't received a Newsletter since early this year? Well, we've been concentrating our efforts on developing the next generation of our product line. New features of muMATH-83 include implicit multiplication, a differential equation solver, a vector algebra and calculus package, hyperbolic trig simplification, an interactive demonstration illustrating the capabilities of each muMATH source file, and an extensively revised manual that includes numerous examples. New features of muLISP-83 include random and sequential file I/O, read and splice macros, a multi-level break debugging facility, a structured, non-local exit mechanism (CATCH and THROW), and an on-line, interactive LISP tutorial system. Contact The Soft Warehouse for details and upgrade discounts.

Recent Reviews

"muMATH," Marketalk Reviews, Softalk for the IBM Personal Computer, August 1983, p. 80. A short favorable review of muMATH-80 for the IBM PC.

"muLISP/muSTAR-80 develops, edits, debugs LISP," John Halamka, InfoWorld, March 14, 1983, p. 54. An "Excellent" and three "Good"s for muLISP-80 for CP/M!

"LISP for CP/M," William G. Wong, Microsystems, August 1983, p. 30. A comparative review of muLISP-80, SuperSoft LISP, and Stiff Upper LISP.

"The Trapdoor Algorithm," David Block, Creative Computing, May 1983, p. 189. The application of muMATH to cryptography problems.

A muMATH User Group

The following people have expressed an interest in establishing an informal muMATH User Group for the exchange of muMATH application programs and user techniques:

Graeme F. Dennes
104 Whippoorwill Drive
Warner Robins, GA 31093
IBM muMATH

Dr. Marjorie Vold
17465 Plaza Animado #144
San Diego, CA 92128
TRS-80 Model 3 muMATH

Wade Ellis Jr.
4562 Alex Drive
San Jose, CA 95130
Apple, SoftCard, and IBM muMATH

John Willis
248 W. Portola Ave.
Los Altos, CA 94022
Apple ADIOS muMATH

Donna Lalonde
Chemistry Department
University of Kansas
Lawrence, KS 66045
8" CP/M muLISP, muMATH

Grahame Wilson
P.O. Box 145
Glebe, N.S.W. 2037
AUSTRALIA
8" CP/M muMATH

Dr. Michael Perelman
Dept. of Economics
California State Univ., Chico
Chico, CA 95929
8" CP/M muMATH
Other economists?

A Power-Series Powering

Professor James Wendel, Dept. of Mathematics,
University of Michigan, Ann Arbor, Michigan, 48109

Although quite general, the muMATH TAYLOR function can be extremely slow because it is unnecessarily time consuming to compute high-order derivatives for mere substitution. Knuth describes more efficient methods in Volume 2 of his book. The file PSPOW.MUS contains an example of the algorithm for raising a truncated power series to an arbitrary power, which need not be integer or even numeric. As a special case, the function can be used to efficiently raise univariate polynomials to integer powers.

The function PSPOW represents the series as a list of successive coefficients, beginning with that of the 0-degree term. If a power series does not begin with 1 as its 0-degree term, then divide the series by its lowest-degree nonzero term and use the resulting "normalized" series. For example, to compute (COS x)^(1/3) to 6th degree, issue the command

? **PSPOW (LIST (1, 0, -1/2, 0, 1/24, 0, -1/720), 1/3, 6) &**

```
% File  PSPOW.MUS              10/05/83              James Wendel %

% Non-integer problems require ARITH.MUS, irrational ALGEBRA.ARI.  %

% PSPOW (v, a, n)  returns a list of the coefficients for the power-
     series expansion of  v^a, truncated to <n>th degree.  <a> is a
     rational number, <n> is a nonnegative integer, and <v> is a
     list of polynomial or truncated power-series coefficients,
     starting with the zeroth degree coefficient, which must be 1
     unless <a> is 0 or 1.  %

FUNCTION PSPOW (V, A, N,
    % Locals: % M, V1, W1, VAL, K),
  WHEN ZERO (A), LIST (1) EXIT,
  WHEN A EQ 1, V EXIT,
  WHEN POP (V) EQ 1,
    A: A + 1,
    W: LIST (1),
    M: 1,
    LOOP
      WHEN M - N EQ 1, REVERSE (W) EXIT,
      V1: V,
      W1: W,
      VAL: 0,
      K: 1,
      LOOP
        WHEN K - M EQ 1 OR EMPTY (V1) OR EMPTY (W1) EXIT,
        VAL: VAL + POP (W1) * POP (V1) * (K * A - M),
        K: K + 1,
      ENDLOOP,
      PUSH (VAL/M, W),
      M: M + 1,
    ENDLOOP EXIT,
ENDFUN$

RDS ()$
```

Collection of Variables

Daniel Bump, 1421 Rosearden Drive,
Forest Grove, Oregon, 97116

Given an expression that is a polynomial in a subset of its variables, it is often desired to collect terms that are similar with respect to this subset and perhaps also to extract the resulting coefficients. The file STFORM.ALG accomplishes this, as illustrated by the dialogue:

```
    ? STVAR: LIST (X, Y)$ % Assign desired subset to global STVAR %
```

```
? STFORM ((X-A)*(X-B)*(Y+A)*(Y+B));
@: A^2 B^2 + (-A B^2 - A^2 B) X + (A B^2 + A^2 B) Y
  + (-2 A B - A^2 - B^2) X Y + (-A - B) X Y^2 + (A + B) X^2 Y
  + X^2 Y^2 + A B X^2 + A B Y^2

? COEFFICIENT (X*Y);
@: - 2 A B - A^2 - B^2

? COEFFICIENT (1);
@ A^2 B^2
```

STFORM and COEFFICIENT communicate via a global variable named COEFLIST, which holds the coefficients most recently produced by STFORM. The argument of STFORM can be nonpolynomial in variables outside STVAR. The result produced by STFORM is not in standard lexical order, so attempts to combine it with other expressions are liable to yield incompletely simplified results.

```
% File  STFORM.ALG            07/07/83       Daniel W. Bump  %

% If APPEND is not primitively defined in your version of muSIMP,
  it must be defined as described in the programming lessons. %

FUNCTION STSPLIT1 (EX1, EX2, LEX,
    % Local: % EX3),
  WHEN EMPTY (LEX),
    ADJOIN (EX1, EX2) EXIT,
  EX3: STSPLIT2 (EX1, 1, POP (LEX)),
  STSPLIT1 (EX1/EX3, EX2*EX3, LEX),
ENDFUN$

FUNCTION STSPLIT2 (EX1, EX2, EX3),
  WHEN FREE (EX1, EX3), EX2 EXIT,
  STSPLIT2 (EX1/EX3, EX2*EX3, EX3),
ENDFUN$

FUNCTION STFILT (LEX1),
    % Refers to outside var STVAR %
  WHEN EMPTY (LEX1), LEX1 EXIT,
  ADJOIN (STSPLIT1 (POP (LEX1), 1, STVAR), STFILT (LEX1)),
ENDFUN$

FUNCTION STSORT (LEX1),
    % Sets outside var COEFLIST %
  COEFLIST: FALSE,
  LOOP
    WHEN EMPTY (LEX1), COEFLIST EXIT,
    COEFLIST: STMERG (POP (LEX1), COEFLIST),
  ENDLOOP,
ENDFUN$

FUNCTION STMERG (EX1, LEX1, % Optional: % LEX2,
    % Local: % EX2),
  WHEN EMPTY (LEX1),
```

```
    REVERSE (LEX2, LIST (EX1)) EXIT,
  WHEN REST (EX1) = REST (EX2: POP (LEX1)),  REVERSE (LEX2,
      ADJOIN (ADJOIN (POP (EX1) + FIRST (EX2), EX1), LEX1)) EXIT,
  WHEN ORDERED (REST (EX1), REST (EX2)),
    REVERSE (LEX2, ADJOIN (EX1, ADJOIN (EX2, LEX1))) EXIT,
  STMERG (EX1, LEX1, ADJOIN (EX2, LEX2)),
ENDFUN$

FUNCTION STUNFILT (LEX1),
  WHEN EMPTY (LEX1), LEX1 EXIT,
  ADJOIN (STUNSPLT (POP (LEX1)), STUNFILT (LEX1)),
ENDFUN$

FUNCTION STUNSPLT (EX1,
    % Local: % EX2),
  WHEN (EX2: POP (EX1)) EQ 1, EX1 EXIT,
  WHEN EX1 EQ 1, EX2 EXIT,
  WHEN PRODUCT (EX2),
    WHEN PRODUCT(EX1),
      ADJOIN ('*, APPEND (REST (EX2), REST (EX1))) EXIT,
    ADJOIN ('*, APPEND (REST (EX2), LIST (EX1))) EXIT,
  WHEN PRODUCT (EX1),
    ADJOIN ('*, ADJOIN (EX2, REST (EX1))) EXIT,
  LIST ('*, EX2, EX1),
ENDFUN$

STVAR: FALSE$

FUNCTION STFORM (EX1),
  WHEN SUM (EX1: EXPAND (EX1)),
    EX1: STUNFILT (STSORT (STFILT (REST (EX1)))),
    WHEN EMPTY (REST (EX1)), FIRST (EX1) EXIT,
    ADJOIN ('+, EX1) EXIT,
  EX1
ENDFUN$

FUNCTION COEFFICIENT (EX1,
    % Local: % EX2, LEX1),
    % Refers to global COEFLIST %
  LEX1: COEFLIST,
  LOOP
    WHEN EMPTY (LEX1), 0 EXIT,
    WHEN REST (EX2: POP (LEX1)) = EX1,
      FIRST (EX2) EXIT,
  ENDLOOP,
ENDFUN$

RDS ()$
```

Symmetric Polynomials

Daniel Bump, 1421 Rosearden Drive,
Forest Grove, Oregon, 97116

Any polynomial that is symmetric in a set of variables R1, R2, .., RN can be expressed as a polynomial function of related variables C1, C2, ..., CN, where

```
C1 = R1 + R2 + ... + RN,
C2 = R1 R2 + R1 R3 + ... + RNM1 RN,
...
CN = R1 R2 ... RN.
```

Using the dummy variable X, these "elementary symmetric polynomials" C1 through CN can be generated by the relation

$$(X-R1)(X-R2) \ldots (X-RN) = X^N - C1\ X^{N-1} + \ldots + (-1)^N\ CN.$$

The relationships are of fundamental algebraic importance. Moreover, symmetric polynomials frequently arise in practical applications, and the problem is often substantially simpler when expressed in this symmetric basis. File SYMMETRY.STF implements an algorithm from Lang's Algebra book for converting from Rs to Cs. The following dialogue computes the discriminant of a cubic:

```
? SYMMETRIC ((R1-R2)^2 * (R2-R3)^2 * (R3-R1)^2);
@: 18 C1 C2 C3 + C1^2 C2^2 - 4 C1^3 C3 - 4 C2^3 - 27 C3^2
```

```
% File  SYMMETRY.STF          07/07/83        Daniel W. Bump %

% Prerequisite file:  STFORM.ALG %

% Uses outside indeterminates R1, R2, ...  and C1, C2, ... %

FUNCTION R(N),
  COMPRESS (LIST ('R, N)),
ENDFUN$

FUNCTION C(N),
  COMPRESS (LIST ('C, N)),
ENDFUN$

FUNCTION SYMMETRIC (P,
    % Local: % N, NUMNUM, DENDEN, DENNUM, NUMDEN, EXPBAS, BASEXP,
               PWREXPD, COEFLIST),
  NUMNUM: DENDEN: DENNUM: EXPBAS: 30,
  BASEXP: -30,
  PWREXPD: 6,
  NUMDEN: N: 0,
  LOOP
    WHEN FREE (P, R (N: N+1)),
      SYMMETRIC1 (P, N-1) EXIT,
```

```
  ENDLOOP,
ENDFUN$

FUNCTION SYMMETRIC1 (P, N,
    % Local: %  P1, P2),
  WHEN N EQ 1,
    EVSUB (P, 'R1, 'C1) EXIT,
  WHEN FREE (P, 'R1), P EXIT,
  P1: SYMMETRIC1 (EVSUB (P, R(N), 0), N - 1),
  P2: SYMMETRIC2 ((P - P1) / C(N), N),
  P1 + C(N) * SYMMETRIC1 (P2, N),
ENDFUN$

FUNCTION SYMMETRIC2 (P, N,
    % Local: %  COUNT, F),
    % Sets outside var STVAR to LIST (outside var X) %
  COUNT: N,
  F: 1,
  LOOP
    WHEN ZERO (COUNT) EXIT,
    F: F * (X + R (COUNT)),
    COUNT: COUNT - 1,
  ENDLOOP,
  STVAR: ADJOIN (X),
  STFORM (F),
  COUNT: N,
  LOOP
    WHEN ZERO (COUNT), P EXIT,
    P: EVSUB (P, C (COUNT), COEFFICIENT (X ^ (N - COUNT))),
    COUNT: COUNT - 1,
  ENDLOOP,
ENDFUN$

RDS ()$
```

Partial Fractions & Other Goodies

Dr. Alan K. Head, CSIRO, Div. of Chem. Physics,
P.O. Box 160, Clayton, Victoria, 3168 AUSTRALIA

File PARFRAC.ALG contains functions for partial fraction decomposition, polynomial quotients and remainders, polynomial greatest common divisors, and cancellation of polynomial GCDs from the numerator and denominator of a rational function.

The DIVOUT function cancels hidden factors having more than one term and containing a specified variable:

```
? DIVOUT ((x^2 + 2*x + 1) / (x^2 - 1),  x);
@: (1 + x)/(-1 + x)
```

The PREM function returns the polynomial remainder of its first argument divided by the second argument, with the third argument considered as the main variable:

```
? PREM (x^3 + 3,  x^2 + 1,  x);
@: 3 - x
```

The PQUOT function returns the quotient corresponding to PREM:

```
? PQUOT (x^3 + 3,  x^2 + 1,  x);
@: x
```

The PGCD function returns the polyomimal gcd of its first two arguments (polynomials) with respect to its third argument (a variable) times a factor that does not depend upon the variable:

```
? FCTR (PGCD (x^2 + 2*x + 1,  x^2 - 1,  x));
@: 2*(1 + x)
```

The PARFRAC function returns a Partial Fraction Expansion, with respect to a given variable, relative to explicit or one-term factors in the denominator:

```
? PARFRAC (1 / ((x + 1)*(x - 1)),  x);
@: 1/(-2 + 2*x) - 1/(2 + 2*x)
```

```
% File PARFRAC.ALG          (c)        12/12/83       Alan K. Head %

FUNCTION DEGCOEF (EX1,
    % Local: % EX2, EX3, LEX1),
    % Fluid from PHIGH:  INDET %
  WHEN FREE (EX1, INDET) EXIT,
  BLOCK
    WHEN PRODUCT (EX1),
      LEX1: REST (EX1) EXIT,
    LEX1: ADJOIN (EX1),
  ENDBLOCK,
  LOOP
    WHEN NOT FREE (EX3: POP (LEX1), INDET),
      WHEN BASE (EX3) = INDET
          AND NUMBER (EX2: EXPON (EX3))
          AND FREE (EX1: EX1/EX3, INDET),
        ADJOIN (EX2, EX1) EXIT EXIT,
  ENDLOOP
ENDFUN$


% PHIGH (EX1, INDET)  returns FALSE if EX1 is not a term or sum of
  terms, each of which is free of INDET or is a numeric power of
  INDET, perhaps multiplied by a coefficient.  Otherwise returns
  (degree, coeff) for the term having the highest power of INDET
  in EX1 %
```

```
FUNCTION PHIGH (EX1, INDET,
    % Local: % EX2, EX3),
  WHEN FREE (EX1: EXPAND (EX1), INDET),
    ADJOIN (0, EX1) EXIT,
  BLOCK
    WHEN SUM (EX1),
      POP (EX1) EXIT,
    EX1: ADJOIN (EX1),
  ENDBLOCK,
  LOOP
    WHEN EMPTY (EX2: DEGCOEF (POP (EX1))), FALSE EXIT,
    BLOCK
      WHEN ATOM (EX2) EXIT,
      WHEN EMPTY (EX3),
        EX3: EX2 EXIT,
      WHEN FIRST (EX2) < FIRST (EX3) EXIT,
      WHEN FIRST (EX2) > FIRST (EX3),
        EX3: EX2 EXIT,
      EX3: ADJOIN (POP (EX2), EX2 + REST (EX3)),
    ENDBLOCK,
    WHEN FREE (EX1, INDET), EX3 EXIT,
  ENDLOOP
ENDFUN$

% PDIV returns (quotient . remainder) of EX1/EX2 wrt INDET %

FUNCTION PDIV (EX1, EX2, INDET,
    % Locals %  EX3, EX4, EX5, EX6, EX7, EX8),
  WHEN FREE (EX2, INDET),
    ADJOIN (EX1 / EX2, 0) EXIT,
  EX3: DEN (EX1: FCTR(EX1)),
  EX1: NUM (EX1),
  EX4: DEN (EX2: FCTR(EX2)),
  WHEN EX5: PHIGH (EX2: NUM (EX2), INDET),
    EX6: 0,
    LOOP
      WHEN EMPTY (EX7:PHIGH(EX1,INDET)),
        ADJOIN (EX1/EX2, 0) EXIT,
      WHEN NEGATIVE (EX8: FIRST (EX7) - FIRST (EX5)),
      ADJOIN (EX6 * EX4 / EX3, EX1 / EX3) EXIT,
      EX3: EX3 * REST (EX5),
      EX6: EX6 * REST (EX5) + INDET ^ EX8 * REST (EX7),
      EX1: EXPD (EX1 * REST (EX5) - INDET ^ EX8 * REST (EX7) * EX2),
    ENDLOOP EXIT,
  ADJOIN (EX1 / EX2, 0),
ENDFUN$

FUNCTION PQUOT (EX1, EX2, INDET),
  FIRST (PDIV (EX1, EX2, INDET)),
ENDFUN$

FUNCTION PREM (EX1, EX2, INDET),
  REST (PDIV (EX1, EX2, INDET)),
ENDFUN$
```

```
% PDCG returns  (gcd(EX1, EX2), COFACTOR (EX1), COFACTOR (EX2)) with
      respect to INDET %

FUNCTION PDCG (EX1, EX2, INDET, EX3,
    % Locals % EX4, EX5, EX6, EX7),
  EX6: EX3,  EX4: EX5: 0,
  LOOP
    WHEN ZERO (REST (EX7: PDIV (EX1, EX2, INDET))),
      LIST (EX2, EX5, EX3) EXIT,
    EX1: EX2,
    EX2: REST (EX7),
    EX3: EX4 - (EX4: EX3) * FIRST (EX7),
    EX5: EX6 - (EX6: EX5) * FIRST (EX7),
  ENDLOOP,
ENDFUN$

% PARFRAC returns the partial fraction expansion of EX1 relative
  to INDET %

FUNCTION PARFRAC (EX1, INDET,
    % Locals % EX2),
  FIRST (EX1: PDIV (NUM (EX1: FCTR (EX1)), EX2: DEN (EX1), INDET))
    + PARF (REST (EX1), EX2),
ENDFUN$

FUNCTION PARF (EX1, EX2,
    % Locals % EX3),
  WHEN ZERO (EX1: REST (PDIV (EX1, EX2: FCTR (EX2), INDET))), 0 EXIT,
  WHEN PRODUCT (EX2),
    WHEN FREE (FIRST
        (EX3: PDCG (EX2/SECOND(EX2), SECOND(EX2), INDET, 1)), INDET),
      PARF (EX1*SECOND(EX3)/FIRST(EX3), SECOND(EX2))
        + PARF (EX1*THIRD(EX3)/FIRST(EX3), EX2/SECOND(EX2)) EXIT,
    'PARFRAC/0 EXIT,
  REST (EX3: PDIV (EX1, BASE(EX2), INDET))
    / EX2 + PARF (FIRST (EX3), EX2 / BASE (EX2)),
ENDFUN$

% PGCD returns the polynomial gcd of EX1 & EX2 wrt INDET %

FUNCTION PGCD (EX1, EX2, INDET),
  FIRST (PDCG (EX1, EX2, INDET, 0)),
ENDFUN$

% DIVOUT (EX1, INDET)  returns EX1 after canceling the gcd of its
     numerator & denominator with respect to INDET %

FUNCTION DIVOUT (EX1, INDET,
    % Locals: % EX2),
  WHEN FREE (EX2:
      NUM (FCTR (PGCD (NUM(EX1:FCTR(EX1)), DEN(EX1),INDET))), INDET),
    EVAL (EX1) EXIT,
  EVAL (FCTR (FIRST (PDIV (NUM(EX1), EX2, INDET))
      / FIRST (PDIV (DEN(EX1), EX2, INDET)))),
ENDFUN$
```

SOUNDEX Codes

Dr. Stanley Schwartz, Department of Pathology,
The Memorial Hospital, Pawtucket, Rhode Island, 02860

Function SOUNDEX in file SOUNDEX.LIB converts names into SOUNDEX codes. A typical use is in keying a large data base of people's names. A SOUNDEX code consists of a four character code, the first of which is the initial letter of the name, and the rest of which are numbers chosen to group letters with similar sounds. If a letter has a zero code or the same code as an adjacent letter, it is dropped. A limitation of the SOUNDEX system is its inability to deal with variant initial letters. Otherwise, surnames with similar sounds or which are variants of the same name (such as Carpenter and Charpentier), usually receive the same SOUNDEX code. A SOUNDEX program written in BASIC appeared in BYTE magazine about two years ago.

```
% File  SOUNDEX.LIB          10/09/83         Dr. Stanley Schwartz %

(DEFUN SOUNDEX (LAMBDA (NAME
    % Local: % LENGTH)
  (SETQ LENGTH T)
  (SETQ NAME (UNPACK NAME))
  (PACK (CONS (CAR NAME) (REVERSE (S-FILL (S-AUX (CDR NAME))) ))) ))

(DEFUN S-AUX (LAMBDA (NAME OUT
    % Local: % TEMP)
  ((NULL NAME) OUT)
  ((EQ (LENGTH OUT) 3) OUT)
  ((ZEROP (SETQ TEMP (S-TRANS (POP NAME) SOUNDEX))) (S-AUX NAME OUT))
  ((EQ TEMP (CAR OUT)) (S-AUX NAME OUT))
  (S-AUX NAME (CONS TEMP OUT)) ))

(DEFUN S-FILL (LAMBDA (LST)
  ((EQ (LENGTH LST) 3) LST)
  (S-FILL (CONS 0 LST)) ))

(DEFUN S-TRANS (LAMBDA (LETTER SOUND)
  ((NULL SOUND) 0)
  ((MEMBER LETTER (CAR SOUND)) (CAAR SOUND))
  (S-TRANS LETTER (CDR SOUND)) ))

(SETQ SOUNDEX (QUOTE ( (0 A E I O U Y H W)
  (1 B F P V)
  (2 C G J K Q S X Z)
  (3 D T)
  (4 L)
  (5 M N)
  (6 R) )))

(RDS)
```

Solving General Relativity Problems

Professor Frederick Ernst at the Illinois Institute of Technology in Chicago has developed a muLISP program for solving problems in general relativity. The article "Charged Spinning-Mass Field Involving Rational Functions" by Chen, Guo and Ernst describe an original application of this program. It appears in the Journal of Mathematical Physics, Vol. 24, No. 6, June 1983, pp. 1564-1567.

Turtle Graphics

Russell Sasamori, Dept. of Electrical Engineering,
University of Hawaii, Honolulu, Hawaii, 96822

The file TURTLE.LIB defines some primitive functions for doing turtle graphics using algorithms described in the book Turtle Graphics by Abelson and deSessa (MIT Press). The file also contains some commented functions illustrating the use of the primitive turtle functions. The back cover of the Newsletter shows the result of a call to the function OHM.

In order to make the functions work on nongraphics console screens, character-level resolution is used and angles are limited to multiples of 45 degrees. Lesson 4 of the interactive, on-line LISP programming lessons, distributed with muLISP-83, indicates how to handle arbitrary angles and provides an example of how to interface to a graphics screen.

TURTLE.LIB requires the function (CLRSCRN) to clear the console screen and the function (CURSOR row col) to position the cursor to <row> and <col> on the screen. If these two functions are not primitively defined in your version of muLISP, you can define them in muLISP by using PRIN1 to send the appropriate sequence of characters to your console screen.

TURTLE.LIB also requires the utility functions ADD1 and SUB1. They are defined in the file UTILITY.LIB.

```
% File  TURTLE.LIB            10/08/83            Russell Sasamori %

% The following assignments set the turtle to point "EAST" at the
  center of a 24 by 80 character screen: %

(SETQ ROW 12)  (SETQ COL 39)  (SETQ DIR 2)

% NEWDRAW clears the screen, centers the turtle on the screen, and
  evaluates its first argument. %

(DEFUN NEWDRAW (NLAMBDA (FUNFORM ROW COL DIR)
  (CLRSCRN)
  (((NULL (SETQ ROW (EVAL ROW))) (SETQ ROW 12)))
  (((NULL (SETQ COL (EVAL COL))) (SETQ COL 39)))
  (((NULL (SETQ DIR (EVAL DIR))) (SETQ DIR 2)))
  (EVAL FUNFORM) ))

% The following assignment establishes "@" as the character to use
  for displaying the path of the turtle: %

(SETQ CH (QUOTE @))

(SETQ NULLSTR (QUOTE ""))

(DEFUN LOCATE (LAMBDA (R C)
  (SETQ ROW R)  (SETQ COL C)
  NULLSTR))

(DEFUN RIGHT (LAMBDA (N)
  ( ((NULL N) (SETQ N 1)) )
  (SETQ DIR (REMAINDER (PLUS DIR N) 8))
  NULLSTR))

(DEFUN LEFT (LAMBDA (N)
  ( ((NULL N) (SETQ N 1)) )
  (SETQ DIR (REMAINDER (DIFFERENCE DIR N) 8))
  NULLSTR))

(DEFUN PLOT (LAMBDA (ROW COL LINELENGTH)
  ((GREATERP ROW 23))
  ((GREATERP COL 79))
  ((LESSP ROW 0))
  ((LESSP COL 0))
  (CURSOR ROW COL)
  (PRIN1 CH)
  NULLSTR))

(DEFUN FORWARD (LAMBDA (N)
  ( ((NULL N) (SETQ N 1)) )
  ((MINUSP N) NULLSTR)
  (SETQ DIR (REMAINDER DIR 8))
  (LOOP
    ( ((EQ DIR 0) (SETQ ROW (SUB1 ROW)))
      ((EQ DIR 1) (SETQ ROW (SUB1 ROW))  (SETQ COL (ADD1 COL)))
      ((EQ DIR 2) (SETQ COL (ADD1 COL)))
```

```
      ((EQ DIR 3) (SETQ ROW (ADD1 ROW))  (SETQ COL (ADD1 COL)))
      ((EQ DIR 4) (SETQ ROW (ADD1 ROW)))
      ((EQ DIR 5) (SETQ ROW (ADD1 ROW))  (SETQ COL (SUB1 COL)))
      ((EQ DIR 6) (SETQ COL (SUB1 COL)))
      (SETQ ROW (SUB1 ROW))  (SETQ COL (SUB1 COL)) )
    (PLOT ROW COL)
    ((ZEROP (SETQ N (SUB1 N))) NULLSTR) ) ))

% A space filling curve function, e.g.  (NEWDRAW (OHM 6) 23 0)  %

(DEFUN OHM (LAMBDA (N)
  ((LESSP N 2) NULLSTR)
  (LEFT 2)  (MHO (SUB1 N))
  (FORWARD 2)  (RIGHT 2)  (OHM (SUB1 N))
  (FORWARD 2) (OHM (SUB1 N))
  (RIGHT 2)  (FORWARD 2)  (MHO (SUB1 N))  (LEFT 2) ))

(DEFUN MHO (LAMBDA (N)
  ((LESSP N 2) NULLSTR)
  (RIGHT 2)  (OHM (SUB1 N))
  (FORWARD 2)  (LEFT 2)  (MHO (SUB1 N))
  (FORWARD 2)  (MHO (SUB1 N))
  (LEFT 2)  (FORWARD 2)  (OHM (SUB1 N))  (RIGHT 2) ))

% A space filling curve function, e.g.  (NEWDRAW (S 2) 0 0)  %

(DEFUN S (LAMBDA (N)
  ((MINUSP N) NULLSTR)
  (S (SUB1 N))
  (FORWARD 2)  (LEFT 2)  (S (SUB1 N))
  (RIGHT 2)  (FORWARD 2)  (S (SUB1 N))
  (RIGHT 2)  (FORWARD 2)  (S (SUB1 N))
  (RIGHT 2)  (FORWARD 2)  (S (SUB1 N))
  (FORWARD 2)  (LEFT 2)  (S (SUB1 N))
  (FORWARD 2)  (LEFT 2)  (S (SUB1 N))
  (FORWARD 2)  (LEFT 2)  (S (SUB1 N))
  (RIGHT 2)  (FORWARD 2)  (S (SUB1 N)) ))

% Pretty crosses, eg:  (NEWDRAW (CROSSES 3) 23 0)  %

(DEFUN CROSSES (LAMBDA (N)
  ((LESSP N 1)
    (FORWARD 3)
    (LEFT 2)
    (FORWARD 3) )
  (CROSSES (SUB1 N))
  (RIGHT 2)  (FORWARD 2)  (RIGHT 2)
  (CROSSES (SUB1 N))
  (FORWARD 2)  (LEFT 2)  (FORWARD 2)
  (CROSSES (SUB1 N))
  (RIGHT 2)  (FORWARD 2)  (RIGHT 2)
  (CROSSES (SUB1 N)) ))

(RDS)
```