N E W S L E T T E R # 1 1

September 1984

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.

Journal of Symbolic Computation

Devoted to computer algebra, automated theorem proving, automatic programming and algorithmic geometry, this new journal is scheduled to begin publication in early 1985. For information about subscriptions or submissions, write Academic Press, 24-28 Oval Road, London, ENGLAND, NW1 7DX.

EUROCAL Computer Algebra Conference

The next international computer algebra conference is EUROCAL 85, Linz Austria, April 1-3, 1985. For information about registration or submissions, write Professor Bruno Buchberger, Universitat Linz, Institut fur Mathematik, A-4040 LINZ, AUSTRIA.

Computer Professionals for Social Responsibility (CPSR)

Many of our muLISP and muMATH users are actively involved in the design and development of sophisticated AI software. In October 1983, DARPA (the Defense Advanced Research Projects Agency of the Defense Department) issued a document entitled "Strategic Computing" which lays out plans for the use of AI and expert systems in our nation's missile defense system. Therefore, we would like to make you aware of CPSR. Their brochure states: "CPSR is a non-profit, tax-exempt, educational organization of computer professionals. It is dedicated to the development and public presentation of expert analyses of society's use of computer technology, particularly as it contributes to the threat of nuclear war." For more information, contact CPSR Inc., P.O. Box 717, Palo Alto, CA, 94301, U.S.A. Phone (415) 322-3778.

Another Comparative LISP Review

The July 1984 issue of BYTE Magazine contains a comparative review of the IBM PC version of muLISP-83 and IQLISP. muLISP does particularly well in the areas of performance benchmarks, memory management, Maclisp and Interlisp compatibility, and tutorials. The authors' implementation of the Sieve of Eratosthenes made both muLISP and IQLISP look bad in comparison to a BASIC implementation. The muLISPer Section of this Newsletter shows the proper way to implement the Sieve of Eratosthenes in Lisp.

* * * * * * * T h e m u M A T H e m a t i c i a n * * * * * *

muMATH User Group

These people have expressed an interest in communicating with other muMATH users for the exchange of application programs and user tips. Send the Newsletter editor your name and address if you want to be listed in the next issue.

Anthony Barcellos, 3000 Cowell Blvd. #205, Davis, CA, 95616
H. Benaroya, Weidlinger Assoc, 333 Seventh Ave, New York, NY, 10001
Harry Bruggesser, Rathausgasse 52, Bern 3011, SWITZERLAND
Dale Bryant, 509 N. Poppy St, Lompoc, CA, 93436
J. Burt, Physics, Atkinson Coll, York U, 4700 Keele St, Downsview, Ont. M3J 1P3 CANADA
Phil Chapman, JPL, 2922 Alta Terrace, La Crescenta, CA, 91214
Paul Duvall, Math, Oklahoma State U, Stillwater, OK, 74078
Dana Eales, Ohio Edison, 76 So. Main, 12th Fl, Akron, OH, 44308
D.J. Herbert, Math&Stats, U.of Pittsburgh, Pittsburgh, PA, 15260
Kjeld Hvatum, C.S. Draper Lab, MS96, 555 Technology Sq, Cambridge, MA, 02139
John Kieft, C.S. Draper Lab, MS11 Rm.5150, 555 Technology Sq, Cambridge, MA, 02139
Roger Kirchner, Math, Carleton Coll, Northfield, MN, 55057
Craig Lindberg, UCSD, A-025, IGPP Bldg. S.I.O, La Jolla, CA, 92037
Joseph Lovett, Sisters of Mercy Health Corp, 28550 Eleven Mile Rd, Farmington Hills, MI, 48018
John Martin, 1114 Lawrence St, Rosenberg, TX, 77471
Thomas Morgan, 202 Argyle Rd, Brooklyn, NY, 11218
Guido Pegna, Dept. di Scienze Fisiche, Universita di Cagliari, Via Ospedale, 72, Cagliari 09100, ITALY
Robert Perry, Physics, U. of Maryland, College Park, MD, 20740
John Robinson, 5908-7 Stevens Forest Rd, Columbia, MD, 21045
David Roper, Physics, Virginia Polytech, Robeson Hall, Blacksburg, VA, 24061
Frank Rumore, Gilian Inst. Corp, No. 8, Dawes Hwy, Wayne, NJ, 07470
Sid Scull, Math, Atkinson Coll, York U, 4700 Keele St, Downsview, Ont. M3J 1P3 CANADA
Michael Singer, Math, North Carolina St. U, Raleigh, NC, 27650
Joachim Wehovz, Calle 207 #4-I-11, Fairview, Rio Piedras, PR, 00926

Implicit Differentiation

The process of finding the derivative of the function defined implicitly by an equation is called implicit differentiation. The derivative can be found by differentiating both sides of the equation and then solving the resulting equation for the derivative. As requested by several users, the file IMPDIF.SOL automates this process and demonstrates its use. It requires both the muMATH SOLVE and DIFferentiation packages.

```
% File  IMPDIF.SOL      (c)   09/08/84    The Soft Warehouse %

FUNCTION IMPDIF (EX1, DYDX),
  SOLVE (DIF1 (EX1, THIRD (DYDX)), DYDX),
ENDFUN $

PROPERTY DIF1, "==", FUNCTION (EX1, EX2),
    % Refers to outside var INDET %
  LIST ('"==", DIF1 (EX1, INDET), DIF1 (EX2, INDET)),
ENDFUN $

COND (WHEN NOT DEMO, RDS () EXIT) $
#ECHO: ECHO $  ECHO: TRUE $

IMPDIF (y(x)^2 + x^2 == r^2,  DIF (y(x), x));
DEPENDS (y(x)) $  #DIFVAR: DIFVAR $  DIFVAR: x $
IMPDIF (y^2 + x^2 == r^2, y`);
DIFVAR: #DIFVAR $ ECHO: #ECHO $
RDS () $
```

muMATH-83 Bug Fixes

If the file ATRG.TRG is dated earlier than 06/05/84, in the function ASIN change the variable name POIN3 to PION3, and change the file date to 06/05/84.

If the file SIGMA.DIF is dated earlier than 06/05/84, in the function SIGMA2M change the name MINUS to NEGATIVE, and change the file date to 06/05/84.

If the file INT.DIF is dated earlier than 06/05/84, in the function RATIONAL change the phrase "RATIONAL (POP (EX1))" to "RATIONAL (POP (EX1), INDET)", and change the file date to 06/05/84.

If the file INTMORE.INT is dated earlier than 09/17/84, in both the properties INTEXPTMS, COS and INTEXPTMS, SIN delete the factor "EX4^(INDET-EX6/EX7)", and change the file date to 09/17/84. This bug was pointed out by John Weinberg of Northrop Corporation who noticed that

```
INT (#E^U COS (A+U), U);
```

returned an incorrect result.

A Simpler Approximate Arithmetic

Prof. Michael Perelman -- California State University, Chico

Prof. Perelman pointed out that a trivial way to truncate an intermediate rational number r to n decimal digits is

```
QUOTIENT (NUM (r * 10^n),  DEN (r)) / 10^n
```

In contrast, the method reported in Newsletter #10 is slower but more accurate, giving the best rational approximation of a given size.

Eigenvalues

Some users have asked how to determine matrix eigenvalues in muMATH. The file EIGENMAT.SOL automates the classic technique and includes a demonstration of its use. Note that approximate numerical methods presented in modern advanced numerical-methods texts are generally preferable when approximate answers are acceptable for totally numeric matrices. Relying upon the limited exact factoring and exact SOLVE routines that muMATH provides, the function EIGENVALS won't return all of the distinct eigenvalues for matrices larger than 4 by 4.

```
% File  EIGENMAT.SOL    (C)   09/06/84    The Soft Warehouse %

FUNCTION EIGENVALS (MAT),
  SOLVE (DET (MAT - 'LAMBDA * IDMAT (LENGTH (MAT) - 1)), 'LAMBDA),
ENDFUN $

COND (WHEN NOT DEMO, RDS () EXIT) $
#ECHO: ECHO $  ECHO: TRUE $

EIGENVALS ({[1, 2], [K, 3]})      % Compute eigenvalues of matrix %;

ECHO: #ECHO $
RDS () $
```

Simplified DOS Commands from muSIMP

The muSIMP EXECUTE command (available if running under MS-DOS 2.0 or later) is quite general since it allows you to execute any DOS function from within muSIMP. However for simple DOS commands, such as DIR, ERASE, or TYPE, it is more complicated than necessary. The following utility function simplifies issuing such DOS commands:

```
FUNCTION DOS (STRNG),
  EXECUTE ("A:COMMAND.COM", COMPRESS (LIST ('"/C ", STRNG))),
ENDFUN$

DOS ("DIR/W")$
```

Naturally, you will need to change the "A:" in the above definition to a different drive letter if your copy of COMMAND.COM is on a drive other than drive A.

Optional Explicit Multiplication

The implicit multiplication provided by muSIMP-83 generally yields more attractive output, but it can mask certain programming or formula input errors, making them hard to discover. Explicit multiplication is also desirable when producing output for use in another programming language that requires asterisks, such as BASIC. The same concerns apply to forcing parentheses around all arguments of unary operators such as SIN and COS.

The following changes to the definition of PRTOPER lets you control these options using the control variables PRTSTAR for explicit/implicit mulitiplication, and PRTPAREN for parenthesizing arguments of unary operators. The simplest way to make the changes is to use the PDS editor to edit the function PRTOPER. Change the 9th line of PRTOPER from

```
          WHEN NOT ATOM (EX2) AND GET ('LBP, FIRST (EX2)),
```

to

```
          WHEN PRTPAREN OR NOT ATOM (EX2) AND GET ('LBP, FIRST (EX2)),
```

Then following the line

```
          WHEN EX1 EQ '*
```

add the line

```
          AND NOT PRTSTAR
```

Once these change are entered, write the new definition of PRTOPER to a file using the PDS WRITE command so the changes can be read back in whenever desired.

Selective Substitution, Expansion & Factoring

Jim Miller -- Laurel, Maryland

Sometimes you may want to substitute, expand, or factor only part of an expression. The file SELECT.ALG implements these useful capabilites and contains a demonstration of their use. Be aware, however, that the resulting expressions may not be canonically ordered, so further operations may not simplify completely.

```
% File SELECT.ALG              03/17/83               Jim Miller %

FUNCTION STRIP (),
    % Modifies outside vars EX1, INT, TLIS1 & TLIS2 %
  TLIS1: LIST (POP (EX1)),
  LOOP INT: INT - 1,  TLIS2: POP (EX1),
       WHEN NOT (INT > 0) EXIT,
       PUSH (TLIS2, TLIS1),
  ENDLOOP
ENDFUN $

FUNCTION SELVU (EX1, INT, % Local: % TLIS1, TLIS2),
  STRIP (),  TLIS2
ENDFUN $

FUNCTION SELEVSUB (EX1, EX2, EX3, INT, % Local: % TLIS1, TLIS2),
  STRIP (),  TLIS2: EVSUB (TLIS2, EX2, EX3),  REPACK (),
ENDFUN $

FUNCTION REPACK (),
  % Refers to outside vars EX1 & TLIS2, Modifies outside var TLIS1 %
  PUSH (TLIS2, TLIS1),  REVERSE (TLIS1, EX1),
ENDFUN $

FUNCTION SELFCTR (EX1, INT, % Local: % TLIS1, TLIS2),
  STRIP (),  TLIS2: FCTR (TLIS2),  REPACK (),
ENDFUN $

FUNCTION SELEXPAND (EX1, INT, % Local: % TLIS1, TLIS2),
  STRIP (),  TLIS2: EXPAND (TLIS2),  REPACK (),
ENDFUN $

FUNCTION SELEXPD (EX1, INT, % Local: % TLIS1, TLIS2),
  STRIP (),  TLIS2: EXPD (TLIS2),  REPACK (),  ENDFUN $

FUNCTION SELTRGX (EX1, INT, TRGEXPD, TRGSQ, % Local: % TLIS1, TLIS2),
  STRIP (),  TLIS2: EVAL (TLIS2),  REPACK (),  ENDFUN $

COND (WHEN NOT DEMO, RDS () EXIT) $  #ECHO: ECHO $  ECHO: TRUE $
SELVU (a + a^2 + a^3, 2)  % select 2nd part % ;
SELEVSUB (a + a^2 + a^3, a, b+c/5, 2)  % substitute in 2nd part % ;
SELEXPAND ((a+b)^2/5 + (c+d)^2/5, 2) % expand 2nd part, distrib den% ;
SELEXPD (COS(1+b/5) + SIN(1+b/5), 2) % expand 2nd part, com den % ;
SELFCTR (COS(a^2+a) + SIN(a^2+a), 2) % factor 2nd part % ;
SELTRGX (COS(2a) + SIN(2a), 2, -3, 0) % 2nd part,TRGEXPD=-3,TRGSQ=0% ;
ECHO: #ECHO $  RDS () $
```

Approximate Inverse Laplace Transform

Dr. Kin-Chu Woo -- Univ. of Alaska, Fairbanks, Alaska

Newsletter #10 contained muSIMP programs for approximate rational arithmetic and for exact Laplace transforms. Exact inverse Laplace transforms are often unobtainable. However, file AINVLAP.ALG implements an approximate numerical method described by Dr. Woo in the October 9, 1983 issue of Electronics (Vol. 53, No. 22, pp. 178ff). The file also contains a demonstration of its use.

```
% File  AINVLAP.ALG           01/03/84       Dr. Kin-Chu Woo %

WTS: LIST (0.0833333333, -32.08333333,  1279.000076, -15623.66689,
            84244.16946,  -236957.5129, 375911.6923, -340071.6923,
            164062.5128, -32812.50256) $

FUNCTION AINVLAP (LAP, TVAL, % Optional: % S,
    % Local: % I, RWTS, ACC),
    % Refers to outside var WTS %
  WHEN POSITIVE (TVAL),
    BLOCK
      WHEN S EXIT,
      S: 'S,
    ENDBLOCK,
    TVAL: 0.69314718/TVAL,
    ACC:I:0,
    RWTS:WTS,
    LOOP
      ACC: ACC + POP (RWTS) * EVSUB (LAP, S, (I:I+1) * TVAL),
      WHEN ATOM (RWTS),
        TVAL * ACC EXIT,
    ENDLOOP EXIT,
ENDFUN $

FUNCTION SHOWAINVLAP (LAP, TINIT, TFINAL, TINC, % Optional: % S),
  LOOP
    WHEN TINIT > TFINAL, '"" EXIT,
    PRTMATH (AINVLAP (LAP, TINIT, S)),
    NEWLINE (),
    TINIT: TINIT + TINC
  ENDLOOP
ENDFUN $

COND (WHEN NOT DEMO, RDS () EXIT) $

#POINT: POINT $  #MAXDEN: MAXDEN $  POINT: 6 $
#ECHO: ECHO $  ECHO: TRUE $
MAXDEN: 10^10  % optional time-saver if approx arith is present % $

SHOWAINVLAP (1/(S^5+S+1), 1, 5, 1) $

POINT: #POINT $  MAXDEN: #MAXDEN $  ECHO: #ECHO $
RDS () $
```

Integer Factorization

Prof. James Wendel -- Univ. of Michigan, Ann Arbor, Michigan

The optional fractional power portion of file ARITH.MUS contains an integer factorization algorithm that uses trial division by a short list of primes. Limiting the maximum trial divisor is sensible for simplification of radicals, but not if you want to factor a number regardless of computing time.

File IFACTOR.MUS contains an algorithm that avoids wasting space on a long list of primes, yet avoids wasted divisions by most non-primes. The file also contains a demonstration of usage. Donald Knuth describes this method and ones for very large numbers in The Art of Computer Programming, Volume 2.

```
% File  IFACTOR.MUS            09/14/84             James Wendel %

%  IFACTOR (N) function returns a list of prime factors in non-
   decreasing order of magnitude of its positive integer argument %

FUNCTION IFACTOR (N, % Locals: % INCS, INCS1, P, D, L),
  % We will divide N by 2, 3, 5 and numbers relatively prime to 30,
      until the divisor P satisfies P * P > N.
    Returns a list of prime factors in nondecreasing order of
      magnitude, so terminate command with "&" rather than ";" %
  WHEN INTEGER (N) AND POSITIVE (N-1),
    INCS: LIST (1, 2, 2, 4, 2, 4, 2, 4, 6, 2, 6),
    INCS1: RREST (INCS),
    P: 2,
    LOOP
      WHEN POSITIVE (P*P-N),
        WHEN N EQ 1, L EXIT,
        ADJOIN (N, L) EXIT,
      LOOP
        D: DIVIDE (N, P),
        WHEN NOT ZERO (REST (D)),
          WHEN ATOM (INCS),
            INCS: INCS1 EXIT EXIT,
        N: FIRST (D),
        PUSH (P, L),
      ENDLOOP,
      P: P + POP (INCS),
    ENDLOOP EXIT,
ENDFUN $

COND (WHEN NOT DEMO, RDS () EXIT) $
#ECHO: ECHO $  ECHO: TRUE $

IFACTOR (45) &

ECHO: #ECHO $
RDS () $
```

muLISP-85 Wish List

The '85 version of muLISP is now midway through the design stage. The major new features being implemented are:

1. Adjustable precision, approximate rational arithmetic. This looks like floating point (e.g. 3.14159) to the user but has much better round-off properties. The numeric functions of muLISP-85 closely follow the Common Lisp standard.

2. Bit mask functions. Primitives are provided for logical bit operations on integers including NOT, AND, IOR, and XOR.

3. New control constructs: IF, PROGN, and RETURN; the automatic elimination of tail recursion.

4. Expanded memory use. muLISP-85 can take advantage of up to 512K bytes of memory, and without the factor of 4 speed degradation experienced by our competitors.

5. Data package system. User data structures, both linked-list and compiled D-code, can be temporarily transferred to RAM memory (external to the 512K) or to a disk file, thus freeing memory.

Now is your big chance to influence the design of the language! If you have any ideas that mesh well with the muLISP design philosophy (i.e. "small is beautiful" and "you can't always get what you want, but you get what you need"), please send them to Al Rich, Applied Logician, The Soft Warehouse.

muLISP User Group

These people have expressed an interest in communicating with other muLISP users for the exchange of application programs and user tips. Send the Newsletter editor your name and address if you want to be listed in the next issue.

Phil Chapman, JPL, 2922 Alta Terrace, La Crescenta, CA, 91214
Brant Cheikes, 18 Lydia Lane, Garden City, NY, 11530
Dana Eales, Ohio Edison, 76 So. Main, 12th Fl, Akron, OH, 44308
David Ginn, Soc.Sec.Admin, 406 Misty Wood Way, Baltimore, MD, 21228
Roger Kirchner, Math, Carleton Coll, Northfield, MN, 55057
Esko Malmberg, Primdata System, Borlange S-78184 SWEDEN
Thomas Morgan, 202 Argyle Rd, Brooklyn, NY, 11218
Guido Pegna, Dept. di Scienze Fisiche, Universita di Cagliari, Via Ospedale, 72, Cagliari 09100, ITALY
Tom Peterson, English, Cal. St, 951 17th St, Santa Monica, CA, 90403
Oscar Revey, 1919 Tenth St, Santa Monica, CA, 90405
Jay Roland, 500 Sloat Ave, Monterey, CA, 93940
Sid Scull, Math, Atkinson Coll, York U, 4700 Keele St, Downsview, Ont. M3J 1P3 CANADA
Joachim Wehovz, Calle 207 #4-I-11, Fairview, Rio Piedras, PR, 00926

Sieve of Eratosthenes

Dr. Stanley Schwartz -- Providence, Rhode Island

The July '84 BYTE review generally finds muLISP to be a highly efficient implementation of LISP. However, the Sieve of Eratosthenes benchmark made muLISP (and IQLISP as well) look much slower than BASIC. This is a fault not of LISP but of the authors' inefficient implementation of the algorithm in LISP.

By taking proper advantage of the language, Stanley Schwartz's implementation of the Sieve of Eratosthenes (with some minor modifications made by The Soft Warehouse) makes a dramatic improvement in the results. On an IBM-PC, the call (SIEVE 8192) takes 157 seconds, which is faster than any version of BASIC reported in either the BYTE magazine articles "A High-Level Language Benchmark" (September 1981, page 180) or "Eratosthenes Revisited" (January 1983, page 283).

```
; File: SIEVE.LIB              09/05/84             Stanley Schwartz


(PUTD (QUOTE DEFUN) (QUOTE (NLAMBDA (NAM$ EXP$)
  (PUTD NAM$ EXP$)
  NAM$ )))

(DEFUN SIEVE (LAMBDA (COUNT
    % Local: % POSITION ARRAY LST)
  (LOOP
    (PUSH T ARRAY)
    ((ZEROP (SETQ COUNT (DIFFERENCE COUNT 1)))
      (SETQ POSITION 3) ) )
  (LOOP
    ((ATOM ARRAY) COUNT)
    ( ((POP ARRAY)
;       (PRIN1 POSITION) (SPACES 1)  ;delete semicolon to see primes
        (SETQ COUNT (PLUS 1 COUNT))
        ((ATOM (SETQ LST (NTH ARRAY POSITION)))
          (SETQ POSITION (PLUS 2 POSITION)) )
        (SETQ POSITION (PLUS 1 POSITION))
        (LOOP
          (RPLACA LST)
          ((ATOM (SETQ LST (NTH LST POSITION)))) )
        (SETQ POSITION (PLUS 1 POSITION)) )
      (SETQ POSITION (PLUS 2 POSITION)) ) ) ))

(RDS)
```

French and Other Accents

A French-Canadian muLISP user wanted his language teaching program to be able to accept vowels with french accents (i.e. è). The IBM-PC extended character set includes such accented vowels but the MS-DOS line edit routine, which is used by muLISP, provides no way to enter them from the keyboard.

To resolve the problem, a function for line editing was written in muLISP. It behaves like a normal line editor except that a vowel immediately followed by an accent character is automatically converted to a single french accent character. The function LINEEDIT is a good example of using muLISP in the "raw input mode". It returns a list of characters entered from the keyboard.

```
; File: LINEEDIT.LIB  (c)         08/31/84        The Soft Warehouse

(PUTD LINEEDIT '(LAMBDA (
     % Local: % CHAR LST READCH RDS )
  (SETQ CHAR (READCH))
  (LOOP
    ((EQ CHAR CR) (REVERSE LST) )
    ( ((EQ CHAR BS)
      ( ((NULL LST))
          (PRIN1 BS)  (SPACES 1)  (PRIN1 BS)  (POP LST) )
        (SETQ CHAR (READCH)) )
      (PRIN1 CHAR)  (PUSH CHAR LST)
      ((MEMBER CHAR VOWELS)
        (SETQ CHAR (READCH))
        ((MEMBER CHAR ACCENTS)
          (SETQ CHAR (GET (CAR LST) CHAR))
          (PRIN1 BS)  (PRIN1 CHAR)  (POP LST)
          (PUSH CHAR LST)  (SETQ CHAR (READCH)) ) )
      (SETQ CHAR (READCH)) ) ) ))

(SETQ CR (ASCII 13))  (SETQ BS (ASCII 8))
(SETQ VOWELS '(a e i o u))
(SETQ ACCENTS (LIST (ASCII 39) ` ^))
(PUT 'a (ASCII 39) (ASCII 160))
(PUT 'a '` (ASCII 133))
(PUT 'a '^ (ASCII 131))
(PUT 'e (ASCII 39) (ASCII 130))
(PUT 'e '` (ASCII 138))
(PUT 'e '^ (ASCII 136))
(PUT 'i (ASCII 39) (ASCII 161))
(PUT 'i '` (ASCII 141))
(PUT 'i '^ (ASCII 140))
(PUT 'o (ASCII 39) (ASCII 162))
(PUT 'o '` (ASCII 149))
(PUT 'o '^ (ASCII 147))
(PUT 'u (ASCII 39) (ASCII 163))
(PUT 'u '` (ASCII 151))
(PUT 'u '^ (ASCII 150))
(RDS)
```

Missionaries & Cannibals Puzzle

Michael Asato -- Uni. of Hawaii, Honolulu, Hawaii

A classic puzzle is to get 3 missionaries and 3 cannibals across a river in a boat that carries 2 people, without the missionaries being eaten whenever they are outnumbered in the boat or on either shore. File MISSCANN.LIB uses a breadth-first graph search technique to solve this puzzle for an arbitrary number of missionaries, cannibals and boat capacity.

In the program a position represents an empty canoe on one of the banks, using a list of the form

(#-Missionaries-on-bank #-Cannibals-on-bank Bank-is-Right)

where Bank-is-Right is T or NIL. A path is a list of successive positions. The solution path is displayed using character "graphics" that will work on any computer screen.

```
; File MISSCANN.LIB           09/13/84             Michael Asato


(PUTD (QUOTE DEFUN) (QUOTE (NLAMBDA (NAM$ EXP$)
  (PUTD NAM$ EXP$)
  NAM$ )))

(DEFUN PLAY (LAMBDA (
    % Local: % TOT-M TOT-C BOAT-SIZE PATH)
  (TERPRI)
  (PRINT "Some missionaries and cannibals want to cross a river in")
  (PRINT "a boat of limited capacity.  Any out-numbered missionaries")
  (PRINT "in the boat or on either bank will be eaten by the")
  (PRINT "otherwise obedient cannibals.  See how to avoid this using")
  (PRINT "the fewest number of crossings:")
  (TERPRI 2)
  (SETQ TOT-M (READ-NUM     "  How many missionaries (1-9)?" 1 9))
  (SETQ TOT-C (READ-NUM     "     How many cannibals (1-9)?" 1 9))
  (SETQ BOAT-SIZE (READ-NUM "How many people in boat (2-4)?" 2 4))
  (SETQ PATH (LIST (LIST TOT-M TOT-C NIL)))
  (SETQ PATH (SHORTEST-PATH (LIST TOT-M TOT-C T) (LIST PATH) PATH))
  ((NULL PATH)
    "No solution exists.")
  (SHOW-PATH PATH) T ))

(DEFUN SHORTEST-PATH (LAMBDA (GOAL QUEUE REACHED
    % Local: % CHILDREN)
  (LOOP
    ((NULL QUEUE) NIL)
    (SETQ CHILDREN (APPLY 'EXPAND (CAAR QUEUE)))
    ((MEMBER GOAL CHILDREN) (REVERSE (CAR QUEUE) (LIST GOAL)))
    (SETQ QUEUE
        (APPEND (CDR QUEUE) (NEW-PATHS CHILDREN (CAR QUEUE)))) ) ))
```

```
(DEFUN NEW-PATHS (LAMBDA (CHILDREN PATH)
  ((NULL CHILDREN) NIL)
  (CONS (CONS (POP CHILDREN) PATH) (NEW-PATHS CHILDREN PATH)) ))

(DEFUN EXPAND (LAMBDA (M C BANK
  % Local: %  M-FOR-BOAT M-WOULD-BE-HERE M-WOULD-BE-THERE C-FOR-BOAT
    C-WOULD-BE-HERE C-WOULD-BE-THERE POSITION CHILDREN)
  % Refers to outside PLAY vars TOT-M TOT-C BOAT-SIZE %
  % Modifies outside SHORTEST-PATH var REACHED %
  (SETQ M-FOR-BOAT M)
  (LOOP
    (SETQ M-WOULD-BE-HERE (DIFFERENCE M M-FOR-BOAT))
    (SETQ M-WOULD-BE-THERE (DIFFERENCE TOT-M M-WOULD-BE-HERE))
    (SETQ C-FOR-BOAT C)
    (LOOP
      (SETQ C-WOULD-BE-HERE (DIFFERENCE C C-FOR-BOAT))
      (SETQ C-WOULD-BE-THERE (DIFFERENCE TOT-C C-WOULD-BE-HERE))
      (AND (LEQUAL 1 (PLUS M-FOR-BOAT C-FOR-BOAT) BOAT-SIZE)
        (NOT-EATEN M-FOR-BOAT C-FOR-BOAT)
        (NOT-EATEN M-WOULD-BE-HERE C-WOULD-BE-HERE)
        (NOT-EATEN M-WOULD-BE-THERE C-WOULD-BE-THERE)
        (SETQ POSITION
            (LIST M-WOULD-BE-THERE C-WOULD-BE-THERE (NOT BANK)))
        (NOT (MEMBER POSITION REACHED))
        (PUSH POSITION CHILDREN)
        (PUSH POSITION REACHED) )
      ((MINUSP (SETQ C-FOR-BOAT (DIFFERENCE C-FOR-BOAT 1)))) )
    ((MINUSP (SETQ M-FOR-BOAT (DIFFERENCE M-FOR-BOAT 1))) CHILDREN) ) ))

(DEFUN NOT-EATEN (LAMBDA (M C)
  ((ZEROP M))
  (LEQUAL C M) ))

(DEFUN SHOW-PATH (LAMBDA (PATH
  % Local: % GAP)
  (SETQ GAP (DIFFERENCE 19 TOT-M TOT-C (QUOTIENT BOAT-SIZE 2)))
  (LOOP
    (LEFT-MIDDLE-RIGHT (CAAR PATH) (CADR (POP PATH))
       (CAAR PATH) (CADAR PATH))
    ((NULL (CDR PATH)) '"")
    (GOING-LEFT (CAAR PATH) (CADR (POP PATH))
       (CAAR PATH) (CADAR PATH)) ) ))

(DEFUN LEFT-MIDDLE-RIGHT (LAMBDA (M C M-NEXT C-NEXT)
  % Refers to outside SHOW-PATH var GAP %
  (TERPRI)  (SHOW-MC M C)  (PRIN1 '|)  (R-BOAT 0 0)
  (SPACES (PLUS GAP GAP))
  (PRIN1 '|)  (SHOW-MC (DIFFERENCE TOT-M M) (DIFFERENCE TOT-C C))
  (TERPRI)
  (SHOW-MC (DIFFERENCE TOT-M M-NEXT) (DIFFERENCE TOT-C C-NEXT))
  (PRIN1 '|) (SPACES GAP)
  (R-BOAT (DIFFERENCE (PLUS M M-NEXT) TOT-M)
          (DIFFERENCE (PLUS C C-NEXT) TOT-C))
  (SPACES GAP)  (PRIN1 '|)
  (SHOW-MC (DIFFERENCE TOT-M M) (DIFFERENCE TOT-C C))
```

```
  (TERPRI)
  (SHOW-MC (DIFFERENCE TOT-M M-NEXT) (DIFFERENCE TOT-C C-NEXT))
  (PRIN1 '|)  (SPACES (PLUS GAP GAP))  (L-BOAT 0 0)  (PRIN1 '|)
  (SHOW-MC M-NEXT C-NEXT) ))

(DEFUN GOING-LEFT (LAMBDA (M C M-NEXT C-NEXT)
  (TERPRI)  (SHOW-MC (DIFFERENCE TOT-M M) (DIFFERENCE TOT-C C))
  (PRIN1 '|)  (SPACES GAP)
  (L-BOAT (DIFFERENCE (PLUS M M-NEXT) TOT-M)
          (DIFFERENCE (PLUS C C-NEXT) TOT-C))
  (SPACES GAP)  (PRIN1 '|)
  (SHOW-MC (DIFFERENCE TOT-M M-NEXT) (DIFFERENCE TOT-C C-NEXT)) ))

(DEFUN SHOW-MC (LAMBDA (M C)
  (REPRIN 'M M)  (SPACES (DIFFERENCE TOT-M M))
  (REPRIN 'C C)  (SPACES (DIFFERENCE TOT-C C)) ))

(DEFUN R-BOAT (LAMBDA (M C)
  (PRIN1 '"[")  (REPRIN 'M M)
  (REPRIN 'C C)  (SPACES (DIFFERENCE BOAT-SIZE M C))  (PRIN1 '>) ))

(DEFUN L-BOAT (LAMBDA (M C)
  (PRIN1 '<)  (SPACES (DIFFERENCE BOAT-SIZE M C))
  (REPRIN 'M M)  (REPRIN 'C C)  (PRIN1 '"]") ))

(DEFUN REPRIN (LAMBDA (PHRASE COUNT)
  (LOOP ((NOT (PLUSP COUNT)) '"")
        (PRIN1 PHRASE)
        (SETQ COUNT (DIFFERENCE COUNT 1)) ) ))

(DEFUN READ-NUM (LAMBDA (PROMPT MIN MAX
  % Local: % ANS READCH)
  (PRIN1 PROMPT)  (SPACES 1)
  (LOOP (SETQ ANS (READCH))
        ((AND (NUMBERP ANS) (LEQUAL MIN ANS MAX)) (PRINT ANS))
        (PRIN1 (ASCII 7)) ) ))

(DEFUN LEQUAL (LAMBDA ARGS
  ((NULL (CDR ARGS)))
  ((LESSP (CAR ARGS) (CADR ARGS)) (APPLY 'LEQUAL (CDR ARGS)))
  ((EQ (CAR ARGS) (CADR ARGS)) (APPLY 'LEQUAL (CDR ARGS))) ))

(PLAY (RDS))
```