N E W S L E T T E R # 1 2

February 1985

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 Soft Warehouse, Inc., P.O. Box 11174, Honolulu, Hawaii, 96828, U.S.A. A complete set of back issues is available on request for $15 ($20 for orders from outside the U.S. and Canada).

SOFT WAREHOUSE, INC.

The Soft Warehouse has celebrated its 6th anniversary by incorporating. With the exception of the name, our address and phone number remain the same. We thank our customers for their loyal support and useful suggestions over the years.

E U R O C A L ' 8 5

The 1985 European Conference on Computer Algebra will be held in Linz, Austria between April 1 - 3, 1985. The conference is being organized by the Organization for Symbolic and Algebraic Manipulation in Europe (SAME) and the ACM Special Interest Group on Symbolic and Algebraic Manipulation (SIGSAM). In addition to papers covering all aspects of symbolic and algebraic computation, there will be demonstrations of various computer algebra systems including muMATH. For information, write Bruno Buchberger, EUROCAL '85, Institut f. Mathematik, Universitat Linz, A-4040 Linz, AUSTRIA, EUROPE; or Telex 2-2323 uni li a.

I J C A I ' 8 5

The Ninth International Joint Conferences on AI (IJCAI) 1985 Conference will be held on the University of California, Los Angeles campus between August 18 - 24, 1985. The conference will be managed by the American Association of Artificial Intelligence (AAAI). In conjunction with the conference, a large exhibit program is scheduled. For information, write AAAI, 445 Burgess Drive, Menlo Park, CA, 94025, U.S.A.; or Telephone (415) 328-3123.

muMATH is continually undergoing improvement. Currently Soft Warehouse, Inc. is shipping version 4.12 of muMATH-83. In addition to a number of bug fixes and internal improvements, version 4.12 extends the class of cubic equations that muMATH can solve. Also the efficiency of the muSIMP compiler and the density of the code it produces has been improved. If you have version 4.xx of muMATH-83 and want to update to 4.12, pack your master disks in an unbendable container and send them to Soft Warehouse, Inc. with $20 ($25 for customers outside the U.S. and Canada). We will update your disks and return them along with approximately 20 pages of manual revisions. Payment must accompany update orders. Money orders, checks, and Visa and Master Card accepted.

muMATH Users

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.

Rex Buddenberg, Naval Postgrad.Sch., SMC #1309, Monterey, CA 93943
Dr. B. Michael Castellano, Math, U.of Alabama, Huntsville, AL 35899
Ed Campbell, 1118 Hudson St., Harrisburg, PA 17104
Robert Crawford, 1141 Morehead Rd, Bowling Green, KY 42101
James Drake, 2971 Shady Hollow E., Boulder, Co 80302
Dr. Charles Dyer, Astronomy, Scarborough Coll, U.of Toronto, 1265 Military Trail, West Hill, Ontario M1C 1A4 CANADA
Dr. Colin Fraser, Math & CS, Dundee Coll.of Tech, 8 Computer Studies, Bell St., Dundee DD1 1HG, SCOTLAND
Dr. Aftab Hassan, George Washington U., 2517 K St. NW #301, Washington, DC 20037-2022
Nobuyuki Inada, Info.Science Lab, Inst.of Phys.&Chem.Res., Hirosawa 2-1, Wako-shi, Saitama 351-01 JAPAN
Richard Joslin, NAMTD 1008, NAS Miramar, San Diego, CA 92145
Al Metzger, 42 Parkholme Ave., Guelph, Ontario N1E 2S3, CANADA
Robert L. Miller, 908 N. Broadway, Apt. 411, Urbana, IL 61801
Dr. Erich Neuwirth, Inst.for Statistics & CS, U.of Vienna, Rathausstr. 19/4, A-1010 Wien, AUSTRIA
Dr. M. Ochiai, Math, Osaka University, Toyonaka, Osaka 560 JAPAN
Dr. T.E. Peacock, Chem., U.of QLD, St. Lucia, QLD 4067, AUSTRALIA
Louis Pecora, Naval Res.Lab, Code 6631, Washington, DC 20375-5000
Michael Saunders, 13 Pioneer Village, Philomath, OR 97370
Ray Shaw, Lively & Assoc., POBox 938, Twentynine Palms, CA 92277
David A. Smith, 2904 Stoneway, Austin, TX 78731
Dr. Gary R. Smith, IBM, 16307 Shady Elms, Houston, TX 77059
Roman Solecki, Mech.Engr., U.of CT, Eng.II Bldg.108A, U-139ME, Storrs, CT 06268
Prof. William Squire, MAE, Coll.of Engr., West Virginia U., Morgantown, WV 26506
William Weller, Math & CS, Shippensburg U, 117 So. Prince St., Shippensburg, PA 17257
Prof. James Wendel, 1505 Hillridge Blvd, Ann Arbor, MI 48103

Polynomial Linear Difference Equations

Dr. Robert Crawford
Bowling Green, Kentucky

Dr. Crawford has contributed a package for polynomial interpolation and for solving polynomial linear difference equations. Here are excerpts from his description:

The main function is FINDPOLY -- the other three are support functions. Given a list L = (x_1, x_2, ..., x_n), a starting location a, and an indeterminant x, FINDPOLY (L, a, x) will produce a polynomial F (x) of minimal degree such that F (a) = x_1, F (a+1) = x_2, ..., and F (a+n-1) = x_n. Usually FINDPOLY needs to be followed by EXPAND or EXPD to affect some simplification. Therefore, the source file ALGEBRA.ARI is a prerequisite for FINDPOLY.ALG.

FINDPOLY is a rather limited function, but still of considerable utility. It does, of course, solve polynomial linear difference equations, which is what I originally designed it for. These may be applied, for example, to estimate the runtime of programs. It may also be easily used to do a polynomial fit where the data points are equally spaced. In general, if y_0, y_1, ..., y_n are the values of a function at b, b+h, ..., b+n h, then set L to LIST (y_0, y_1, ..., y_n) and call FINDPOLY (L, 0, (x-b)/h) to get a polynomial function F (x) such that F (b + i h) = y_i. Examples are included at the end of the file.

```
% File  FINDPOLY.ALG           01/07/85          Robert Crawford %

         % Polynomial Linear Difference Equations Package %

FUNCTION FINDPOLY (L, A, X,
    % Local: % I, F),
  F: FIRST (L),  I: 0,
  LOOP WHEN ISCONSTANT (L), F EXIT,
       I: I + 1,  L: DIFF (L),  F: F + FIRST(L) * C(X-A, I) ENDLOOP
ENDFUN$

FUNCTION ISCONSTANT (L),
    % Determines if the list L is constant %
  LOOP WHEN EMPTY (REST (L)) EXIT,
       WHEN NOT POP (L) = FIRST (L), FALSE EXIT ENDLOOP
ENDFUN$

FUNCTION DIFF (L),
    % Computes the first forward difference of the list L %
  WHEN REST (L), ADJOIN (SECOND (L) - FIRST (L), DIFF (REST (L))) EXIT
ENDFUN$

FUNCTION C (N, K, % Local: % I, T),
    % Computes binomial coefficients %
  T: 1,  I: 0,
  LOOP WHEN I = K,  T/K! EXIT,
```

```
       T: T*(N - I),  I: I + 1 ENDLOOP
ENDFUN$

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

EXPAND (FINDPOLY ('(2 3 5 7), 1, X));
EXPD (FINDPOLY (LIST (1/2, 1, 4/3, 1),  0,  3 (x-2)));

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

SUPER-CALCULATOR Fix

Dr. Erich Neuwirth
Vienna, Austria

Stuart Edwards' SUPER-CALCULATOR units conversion program is described in issue number 8 of the Soft Warehouse Newsletter. Dr. Neuwirth found a bug in the function "in" that caused the result of expressions such as

```
year in minute;
```

to be displayed as

```
6.5743596 10^7 minute/125
```

rather than

```
5.2594876 10^5 minute
```

The following definition of the function "in" corrects this problem:

```
SUBROUTINE in (ex1, units),
  WHEN SECOND (units) EQ 'oF,
    LIST ('oF, EVAL (EVAL (ex1)) / kelvin 9/5 - 459688/1000) EXIT,
  WHEN SECOND (units) EQ 'oC,
    LIST ('oC, EVAL (EVAL (ex1)) / kelvin - 27316/100) EXIT,
  WHEN units EQ 'kelvin,  EVAL (EVAL (EX1)) EXIT,
  EX1: EVAL (EVAL (ex1)) / EVAL (units),
  WHEN ATOM (units),
    WHEN FIRST (EX1) EQ '*,  APPEND (EX1, LIST (units)) EXIT,
    LIST ('*, EX1, units) EXIT,
  WHEN FIRST (EX1) EQ '*,  APPEND (EX1, tihs (units)) EXIT,
  ADJOIN ('*, ADJOIN (EX1, tihs (units)))
ENDSUB$
```

A printed listing of the source and documentation for the SUPER-CALCULATOR package is still available from Soft Warehouse, Inc. Send a check or money order for $10 ($15 outside the U.S.A. and Canada) to Soft Warehouse, Inc. Sorry, no COD or purchase

orders accepted.

Taylor Series Solution of ODEs

muMATH-83 can determine closed form solutions for a great variety of ordinary differential equations. However, closed-form solutions don't always exist, and without strong restrictions on the class of equations and solutions, no known algorithm can guaranteeably find existing solutions. Truncated series solutions are often useful in such situations, and Taylor-series solutions are particularly easy to automate.

One way to accomplish this for first-order equations of the form $y` = f(x,y)$ is to differentiate the equation, then substitute f for the resulting y` in the right side, giving y`` in terms of x and y. Such differentiation and substitution is repeated until we reach the desired order of approximation. We then substitute the initial conditions y_o for y and x_o for x in the right sides to obtain the initial values of these derivatives, which are then used in the truncated series:

$$y = y_o + (x-x_o)\, y`|_{x_o} + \ldots + (x-x_o)^n\, y^{(n)}|_{x_o} / n!$$

File TAYODE.DIF implements this technique together with a demonstration of its use. The method can be extended to systems of first-order equations and to higher-order equations.

```
% File  TAYODE.DIF       (C)   01/01/84      Soft Warehouse, Inc. %

FUNCTION TAYODE (DYDX, XVAR, YOFX, X0, Y0, MAXDEG,
    % Local: % NUMNUM, DENNUM, POWEXPD, DN, DEG, ANS),
  NUMNUM: DENNUM: 6,  POWEXPD: 0,  DN: DYDX,  DEG: 2,
  ANS: Y0 + (XVAR - X0) * EVSUB (EVSUB (DYDX, YOFX, Y0), XVAR, X0),
  LOOP
    WHEN DEG > MAXDEG,  ANS EXIT,
    DN: EVSUB (DIF (DN, XVAR), DIF (YOFX, XVAR), DYDX),
    ANS: ANS + (XVAR-X0)^DEG * EVSUB (EVSUB (DN, YOFX, Y0), XVAR, X0) / DEG!,
    DEG: DEG + 1,
  ENDLOOP
ENDFUN $

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

% Example: 4th degree approximate solution for

                      y` = #E^(x y),  y(0) = 1: %

TAYODE (#E ^ (X*Y(X)), X, Y(X), 0, 1, 4);

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

muLISP Users

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.

Dr. Michael Anbar, 181 Halwill Dr., Snyder, NY 14226

Peter Baumer, Protecon', 1147 E. Broadway #53, Glendale, CA 91205

David Cain, Electric Power Res., POBox 10412, Palo Alto, CA 94303

Ed Campbell, 1118 Hudson St., Harrisburg, PA 17104

Robert Crawford, 1141 Morehead Rd, Bowling Green, KY 42101

Martin Crossland, 6270 South Field Ct., Littleton, CO 80123

Joan Horvath, JPL, 125/224, 4800 Oak Grove Dr., Pasadena, CA 91109

Dr. Gerald Isaacs, Carroll Coll, 100 N. East Ave., Waukesha, WI 53186

Dr. Frank Lin, #10 Stony Hill Village, Brookfield Center, CT 06805

Kenneth Lui, 17331 Poplar St., Fountain Valley, CA 92708

Lawrence Massey, Pansophic Sys., 3720 Longview Dr., Atlanta, GA 30341

Joyce McDowell, 1341 Rocking Horse Ln, La Habra, CA 90631

Robert R. McKenzie, 2518 England St. #1, Huntington Beach, CA 92648

Dr. M. Ochiai, Math, Osaka University, Toyonaka, Osaka 560 JAPAN

Dr. G. Schoffa, Biophysics, U.of Karlsruhe, Engesserstr. 7, Postfach 6380, 75 Karlsruhe 1, WEST GERMANY

Dick Schuller, Battelle, 4000 NE 41st, Seattle, WA 98105

Ray Shaw, Lively & Assoc., POBox 938, Twentynine Palms, CA 92277

G. Timothy Stump, 434 W. Huntington Dr., Arcadia, CA 91006

Julian Taylor, Weserstr. 11, 6070 Langen, WEST GERMANY

Mostafa Terrab, Bechtel, 2000 Broadway Apt. 712, San Francisco, CA 94115

M u s i c E a s e

An Automatic Bass Accompaniment Creator

MusicEase is a software program for the IBM PC that will automatically create a bass accompaniment for your melodies. You create the melody and **MusicEase** creates a full bass clef accompaniment --- all automatically. You just designate the type of accompaniment you want. For example, Swing Bass, Rock, Calypso, or even Hasidic. These are just a few of the predefined styles that come with **MusicEase**.

Then, using SongWright, you can print out professional quality lead sheets on your graphics printer. Or you can edit the result first. Even better, **MusicEase** is customizable. So if you want to create an accompaniment style of your own, it is easily done with **MusicEase**. Once you have designed an accompaniment style, it can be applied to a limitless number of melodies with just a few simple keystrokes.

MusicEase features:

* Automatic composition of bass clef accompaniment.
* Many predefined accompaniment styles.
* Ability to define new accompaniment styles of your own.
* Chords handled include dim7, aug, min, 7, min7, maj7, 6, min6, sus4, -5.

MusicEase requires:

* An IBM PC or compatible computer.
* 128K bytes of random access memory.
* An IBM (or work-alike), Epson or Gemini dot-matrix graphics printer.
* A monochrome or color monitor (a graphics adapter is not necessary).
* One disk drive.
* DOS version 1.1 or above.
* SongWright (Version 2.0).

MusicEase is priced at $95.95, including shipping ($110.00 US outside USA and Canada). **MusicEase** and **SongWright** together are priced at $135.95 ($150.00 US outside USA and Canada). Washington residents please add 7.8% sales tax. To order, mail your check or money order to

Grandmaster, Inc.
P. O. Box 2567
Spokane, WA 99220-2567

Sorry, no COD or charge account orders accepted.

M a x F A C T S

An Expert System Builder/Inference Engine

MaxFACTS is a generic Expert System tool that allows you to create working Expert Systems for any application. It provides all the facilities required to achieve great flexibility including Backward/Forward chaining and Meta-level control. The Meta-level control feature allows you to take control of the inference process so you can fit your Expert System to the problem, not the problem to the Expert System! This control allows you to change validation methods, mix both Backward and Forward chaining (in any amounts), and use the user query interface to obtain relevant data during processing. MaxFACTS's versatile internal pattern matching routines allow you to implement powerful concepts such as recursive code and "Frame" representation of the knowledge base.

Additional features for increased performance are:

Caching: The automatic storing of inferred data for future use. Result: increased speed.

Justification: MaxFACTS can store its processing steps to allow verification of solutions. Result: increased user confidence, solution documentation.

Knowledge Partitioning: This concept allows you to group your Knowledge Base in smaller, more convenient, and easily managed partitions. Result: easy management during both processing and development.

Software Hooks: MaxFACTS provides a facility for you to access special purpose Lisp routines during processing. Result: increased range of problem solving.

Full Indexing: All rules encoded are cross indexed to cause "intelligent" internal processing. Result: increased speed and efficiency.

Price: $175.00 Full MaxFACTS package. Run with full muLISP.
$150.00 Stand alone MaxFACTS package (includes muLISP runtime system). Benefits of Software Hooks lost.

Write: Robert R. McKenzie
2518 England St. #1
Huntington Beach, CA 92648

The Eights Puzzle

Peter Harada -- Honolulu, Hawaii

The eights puzzle consists of eight square tiles numbered 1 through 8, in a square tray having a surface area that accommodates exactly nine tiles. To start, the tiles are placed on the tray in a scrambled order, leaving one empty space. The prime objective is to rearrange the tiles into the order

```
 -----------
| 1 | 2 | 3 |
|---+---+---|
| 8 |   | 4 |
|---+---+---|
| 7 | 6 | 5 |
 -----------
```

by a sequence of sliding tiles vertically or horizontally into the adjacent empty space, without moving any tiles out of the tray. A secondary objective is to reach the goal in a minimum number of moves. The final configuration can be reached from only half of the possible initial configurations. EIGHTS.LIB determines if the final configuration is reachable by a quick direct test entailing even versus odd permutations.

This puzzle is a classic example of problem solving, described in many textbooks and papers on artificial intelligence. The successive board configurations can be regarded as nodes in a graph, with the objective being to search for a path (preferably a shortest one) from the initial to the goal node. The search entails maintaining a list of all visited positions so that paths won't contain wasteful loops -- and also maintaining a list of the active "fringe" nodes which are candidates as points of departure for further graph growing and evaluation. "Breadth-first" search always extends one of the shortest partial paths, which guarantees a shortest path to the goal if memory or patience is not exhausted by maintaining excessively long visited and fringe position lists.

The heuristic search in EIGHTS.LIB expands one of the fringe nodes for which a heuristic estimate of the total path length is least. This estimate is the known partial path length to the fringe node plus an estimate of the minimum remaining distance from that node to the goal node. If the estimates were always perfect, the search would find the shortest path with no wasted exploration. Any consistent nonzero underestimate of the remaining distance can reduce the number of explored nodes while still guaranteeing a shortest path.

As its estimate of the remaining number of moves, the program uses the "taxi metric" plus three times the relative "sequence score". The former is the sum over the eight numbered tiles of the absolute East/West displacement plus the absolute North/South displacement of the tile from its goal position. The latter is 1 if the central square is occupied, plus the sum over the eight noncentral squares of 2 for every tile that is not immediately

followed in clockwise order by its goal successor. This heuristic is not actually an underestimate of the remaining number of moves, so the resulting path is not guaranteed to be a shortest one. Can you discover such an example? Can you discover better heuristics? Can you modify the representations of position, allowable moves or position lists to make the program substantially faster? How about the analogous "fifteens" puzzle, which has a 4 by 4 tray?

```
; File EIGHTS.LIB         02/08/85             Peter A. Harada

(PUTD DEFUN '(NLAMBDA (FUN$ EXP$)  (PUTD FUN$ EXP$)  FUN$ ))

; EIGHTS searches for the next node using an Ordered State-space
; search as described in "Principles of Artificial Intelligence",
; by Nils J. Nilsson, Tioga Publishing Co., Palo Alto, CA, 1980.

(DEFUN EIGHTS (LAMBDA (% Local: % LEGAL-LST RSLT-LST CHAR)
  (LOOP
    (CLRSCRN)  (TERPRI)
    (CENTER "* * *   T h e   E I G H T S   P u z z l e   * * *")
    (CURSOR 3 0)  (TERPRI)
    (CENTER " ----------- ")
    (CENTER "|   |   |   |")
    (CENTER "|---+---+---|")
    (CENTER "|   |   |   |")
    (CENTER "|---+---+---|")
    (CENTER "|   |   |   |")
    (CENTER " ----------- ")
    (SETQ LEGAL-LST (LIST 1 2 3 4 5 6 7 8 " "))
    (SETQ RSLT-LST (LIST NIL NIL NIL NIL NIL NIL NIL NIL NIL))
    (SETQ COL (QUOTIENT (DIFFERENCE (LINELENGTH) 17) 2))
    (READ-ROW 5 COL '(0 1 2))
    (READ-ROW 7 COL '(7 8 3))
    (READ-ROW 9 COL '(6 5))
    (CURSOR 9 (PLUS COL 8))
    (SETQ CHAR (CAR LEGAL-LST))
    ( ((EQ (PRINT CHAR) '" ")
        (SETQ CHAR 9) ) )
    (TERPRI 2)
    (RPLACA (NTHCDR 4 RSLT-LST) CHAR)
    (EIGHTS-AUX RSLT-LST)
    ((NOT (Y-OR-N-P "Do you want me to solve another one"))) ) ))

(DEFUN EIGHTS-AUX (LAMBDA (START GOAL OPEN CLOSED NODE0 NODEI
    NODEJ SUCCESSORS NEXPANSIONS N TEMPOPEN)
  ((NOT (SOLVABLE START))
    (PRINT "That is an impossible starting position.") )
  (SETQ NODE0 START)
  (SETQ GOAL '(1 2 3 4 5 6 7 8 9))
  (SETQ N (SETQ NEXPANSIONS 0))
  (SETQ OPEN (LIST (LIST (F* NODE0 N) N NODE0 NIL)))
  (LOOP
    (PRIN1 '*)
    (SETQ OPEN (APPEND TEMPOPEN OPEN))
```

```
    (SETQ OPEN (DELETE (SETQ NODEI (GET-MIN-NODE OPEN GOAL)) OPEN))
    (PUSH NODEI CLOSED)
    ((EQUAL (CADDR NODEI) GOAL)
      (OUTPUT-SOLUTION START NODEI CLOSED NEXPANSIONS) )
    (SETQ SUCCESSORS (EXPAND-NODE (CADDR NODEI)))
    (SETQ N (PLUS (CADR NODEI) 1))
    (SETQ TEMPOPEN NIL)
    (LOOP
      ((NULL SUCCESSORS))
      (SETQ NEXPANSIONS (PLUS NEXPANSIONS 1))
      (SETQ NODEJ (LIST (F* (CAR SUCCESSORS) N) N (CAR SUCCESSORS)
                  (CADDR NODEI)))
      (SETQ INOPEN (CONTAINED-INP OPEN NODEJ))
      (SETQ INCLOSED (CONTAINED-INP CLOSED NODEJ))
      ( ((AND (NULL INOPEN) (NULL INCLOSED))
          (PUSH NODEJ TEMPOPEN) )
        ((AND (NULL INOPEN) (LESSP (CAR NODEJ) (CAR INCLOSED)))
          (SETQ CLOSED (DELETE (GET-MIN-NODE CLOSED (CADDR NODEJ)) CLOSED))
          (PUSH NODEJ TEMPOPEN) )
        ((AND (NULL INCLOSED) (LESSP (CAR NODEJ) (CAR INOPEN)))
          (SETQ OPEN (DELETE (GET-MIN-NODE OPEN (CADDR NODEJ)) OPEN))
          (PUSH NODEJ TEMPOPEN) ) )
      (POP SUCCESSORS) ) ) ))

(DEFUN CONTAINED-INP (LAMBDA (LST NODE TEMP)
  (SETQ NODE (CADDR NODE))
  (LOOP ((NULL LST) NIL)
        (SETQ TEMP (POP LST))
        ((EQUAL (CADDR TEMP) NODE) TEMP) ) ))

(DEFUN EXPAND-NODE (LAMBDA (NODE)
  (MAPCAR 'XCHG-POS (NTH (POSITION 9 NODE) MOVE-LIST)) ))

(DEFUN XCHG-POS (LAMBDA (PAIR LST TEMP1 TEMP2)
  (SETQ LST (APPEND NODE))
  (SETQ TEMP1 (NTHCDR (CAR PAIR) LST))
  (SETQ TEMP2 (NTHCDR (CADR PAIR) LST))
  (SETQ TEMP (CAR TEMP2))
  (RPLACA TEMP2 (CAR TEMP1))
  (RPLACA TEMP1 TEMP)  LST ))

(SETQ MOVE-LIST '(((0 1) (0 7))          ((0 1) (1 2) (1 8))
              ((1 2) (2 3))        ((3 8) (2 3) (3 4))
              ((4 5) (3 4))        ((5 6) (4 5) (5 8))
              ((5 6) (6 7))        ((7 8) (0 7) (6 7))
              ((7 8) (3 8) (1 8) (5 8)) ) )

(SETQ POS-LIST '((0 1 2 3 4 3 2 1 0)      (1 0 1 2 3 2 3 2 0)
            (2 1 0 1 2 3 4 3 0)    (3 2 1 0 1 2 3 2 0)
            (4 3 2 1 0 1 2 3 0)    (3 2 3 2 1 0 1 2 0)
            (2 3 4 3 2 1 0 1 0)    (1 2 3 2 3 2 1 0 0)
            (2 1 2 1 2 1 2 1 0) ) )

(DEFUN F* (LAMBDA (LAYOUT N)
  (PLUS N (H* LAYOUT N)) ))
```

```
(DEFUN H* (LAMBDA (LAYOUT N PWEIGHT SWEIGHT)
  (SETQ PWEIGHT 1)  (SETQ SWEIGHT 3)            ; Adjustable weights
  (PLUS (TIMES PWEIGHT (P* LAYOUT N))(TIMES SWEIGHT (S* LAYOUT N)))))

(DEFUN P* (LAMBDA (LAYOUT N SUM POSITION)
  (SETQ SUM 0)
  (SETQ POSITION POS-LIST)
  (LOOP
    ((NULL LAYOUT) SUM)
    (SETQ SUM (PLUS SUM
      (NTH (PLUS (POP LAYOUT) -1) (POP POSITION)))) ) ))

(DEFUN S* (LAMBDA (LAYOUT N % Local: % SUM NUM)
  (SETQ SUM 1)
  ( ((EQ (CAR LAYOUT) 9)
      (SETQ LAYOUT (APPEND (CDR LAYOUT))) )
    ((EQ (CAR (LAST LAYOUT)) 9)
      (SETQ SUM 0)
      (SETQ LAYOUT (APPEND LAYOUT)) )
    (SETQ LAYOUT (REMOVE 9 LAYOUT)) )
  (RPLACA (LAST LAYOUT) (CAR LAYOUT))
  (LOOP
    (SETQ NUM (POP LAYOUT))
    ((NULL LAYOUT) SUM)
    ( ((EQ NUM 8)
        ((EQ (CAR LAYOUT) 1))
        (SETQ SUM (PLUS SUM 2)) )
      ((EQ (PLUS NUM 1) (CAR LAYOUT)))
      (SETQ SUM (PLUS SUM 2)) ) ) ))

(DEFUN GET-MIN-NODE (LAMBDA (NODE-LST CONSTANT % Local: % MIN-NODE)
  (SETQ MIN-NODE (POP NODE-LST))
  (LOOP
    ((NULL NODE-LST) MIN-NODE)
    (SETQ NODE (POP NODE-LST))
    ((EQUAL (CADDR NODE) CONSTANT) NODE)
    ( ((GREATERP (CAR MIN-NODE) (CAR NODE))
        (SETQ MIN-NODE NODE) )
      ((AND (EQUAL (CAR MIN-NODE) (CAR NODE))
          (GREATERP (CADR MIN-NODE) (CADR NODE)) )
        (SETQ MIN-NODE NODE) ) ) ))

; SOLVABLE's algorithm is from "Mathematical Games and Pastimes"
; by A. P. Domoryad; Macmillan Co., 1964, Pgs 79-85.

(DEFUN SOLVABLE (LAMBDA (LST % Local: % FLAG)
  (MAPC '(LAMBDA (NUM) (DISORDER NUM LST)) LST)
  (EQ (NOT FLAG) (EVENP (POSITION 9 LST))) ))

(DEFUN DISORDER (LAMBDA (NUM LST)
  ((EQ NUM (CAR LST)))
  ((GREATERP (CAR LST) NUM)
    (SETQ FLAG (NOT FLAG))
    (DISORDER NUM (CDR LST)) )
```

```
  (DISORDER NUM (CDR LST)) ))

(DEFUN OUTPUT-SOLUTION (LAMBDA (START LASTNODE LST EXPANSIONS
    LST1 PRINTLIST)
  (TERPRI 2)
  (PRIN1 "The number of moves checked was ")  (PRIN1 EXPANSIONS)
  (PRINT " and the shortest sequence is:")  (TERPRI)
  (LOOP
    (PUSH (CADDR LASTNODE) PRINTLIST)
    (SETQ LST1 LST)
    (LOOP
      ((EQUAL (CAR (CDDDR LASTNODE)) (CADDR (CAR LST1)))
        (SETQ LASTNODE (CAR LST1)) )
      (POP LST1) )
    ((NULL (CAR (CDDDR LASTNODE)))
      (FORMAT-OUTPUT (PUSH START PRINTLIST)) ) ) ))

(DEFUN FORMAT-OUTPUT (LAMBDA (ANSWERS DISPLAY# S1 S2 S3 COUNTER)
  (SETQ DISPLAY# (QUOTIENT (PLUS 2 (LINELENGTH)) 7))
  (LOOP                        ; Prints DISPLAY# of squares
    ((NULL (CAR ANSWERS)))     ; across the page from a list
    (SETQ COUNTER 0)           ; of lists in ANSWERS
    (SETQ S1)
    (LOOP
      ((EQUAL COUNTER DISPLAY#))
      (SETQ S2 (POP ANSWERS))
      ((NULL S2))
      (PUSH S2 S1)
      (SETQ COUNTER (PLUS COUNTER 1)) )
    (SETQ DISPLAY# COUNTER)
    (SETQ S1 (REVERSE S1))
    (PRINT-ROW S1 DISPLAY# '(0 1 2))
    (PRINT-ROW S1 DISPLAY# '(7 8 3))
    (PRINT-ROW S1 DISPLAY# '(6 5 4))
    (TERPRI) ) ))

(DEFUN PRINT-ROW (LAMBDA (LST DISPLAY# INDEX-LIST)
  (LOOP
    ((ZEROP DISPLAY#) (TERPRI))
    (PRIN3 (NTH (CAR INDEX-LIST) (CAR LST)))
    (PRIN3 (NTH (CADR INDEX-LIST) (CAR LST)))
    (PRIN3 (NTH (CADDR INDEX-LIST) (POP LST)))
    (SPACES 1)
    (SETQ DISPLAY# (PLUS DISPLAY# -1)) ) ))

(DEFUN PRIN3 (LAMBDA (NUM)
  ((EQ NUM 9)  (PRIN1 "  ") )
  (PRIN1 NUM)  (SPACES 1) ))

(DEFUN READ-ROW (LAMBDA (ROW COL LST % Local: % READCH)
  (LOOP
    (CURSOR ROW COL)
    (LOOP
      (SETQ CHAR (READCH))
      ((EQ CHAR (ASCII 27)) (THROW) )     ;Abort EIGHTS if <ESC> typed
```

```
      ((MEMBER CHAR LEGAL-LST))
      (PRIN1 (ASCII 7)) )
    (SETQ LEGAL-LST (REMOVE CHAR LEGAL-LST))
    ( ((EQ (PRIN1 CHAR) '" ")
      (SETQ CHAR 9) ) )
    (RPLACA (NTHCDR (POP LST) RSLT-LST) CHAR)
    ((NULL LST))
    (SETQ COL (PLUS COL 4)) ) ))

(DEFUN CENTER (LAMBDA (MSG)
  (SPACES (QUOTIENT (DIFFERENCE (LINELENGTH) (LENGTH MSG) 8) 2))
  (PRINT MSG) ))

(DEFUN Y-OR-N-P (LAMBDA (QUESTION CHAR READ READCH)
  (PRIN1 QUESTION)
  (PRIN1 "? (Y/N): ")
  (LOOP (SETQ CHAR (READCH))
        ((EQ CHAR 'Y) (PRINT CHAR) T)
        ((EQ CHAR 'N) (PRINT CHAR) NIL)
        (PRIN1 (ASCII 7)) ) ))

(DEFUN POSITION (LAMBDA (ITEM LST % Local: % NUM)
  (SETQ NUM 0)
  (LOOP ((NULL LST) NIL)
        ((EQ ITEM (POP LST)) NUM)
        (SETQ NUM (PLUS NUM 1)) ) ))

(DEFUN NTHCDR (LAMBDA (NUM LST)
  (NTH LST (PLUS NUM 1)) ))

(DEFUN REMOVE (LAMBDA (ITEM LST)
  ((ATOM LST) NIL)
  ((EQUAL ITEM (CAR LST))
    (REMOVE ITEM (CDR LST)) )
  (CONS (CAR LST) (REMOVE ITEM (CDR LST))) ))

(DEFUN DELETE (LAMBDA (ITEM LST % Local: % RSLT)
  ((NULL LST) NIL)
  (LOOP ((NOT (EQUAL ITEM (CAR LST))))
        (POP LST)
        ((NULL LST)) )
  ((NULL LST) NIL)
  (SETQ RSLT LST)
  (LOOP ((NULL (CDR LST)) RSLT)
        ( ((EQUAL ITEM (CADR LST))
          (RPLACD LST (CDDR LST)) ) )
        (POP LST) ) ))

(DEFUN MAPC (LAMBDA (FUN$ LST$)
  (LOOP ((NULL LST$))
        (FUN$ (POP LST$)) ) ))

(DEFUN MAPCAR (LAMBDA (FUN$ LST$)
  ((NULL LST$) NIL)
  (CONS (FUN$ (CAR LST$)) (MAPCAR FUN$ (CDR LST$))) ))        (RDS)
```