```
'####################################################################
'# SOMA figure solver
'# Thorleif Bundgaard <bundgard@post8.tele.dk>
'#    Program user interface.
'# Courtney McFarren  <abcmcfarren@worldnet.att.net>
'#    Solution finder algorithm [ FindSolve: ]
'#
'# V0.1  19.01.1999     First draft using solver routine
'# V0.2  20.01.1999     Structure buildup
'# V0.3  22.01.1999     Various
'# V0.4  04.02.1999     User interface operates 'On demand' from F keys
'# V0.5  05.02.1999     Manual figure input and Batch run
'# V0.6  09.02.1999     BMP and Batch functions
'# V0.7  09.02.1999     Improved manual entry and BMP save
'# V0.70 09.02.1999 TB  Patched the revised ANSWER5 code, added a few comments
'# V0.71 11.02.1999 TB  Small revisiond for stability and display
'# V0.72 15.02.1999 CMF Converted all CUB file references to S(,,)
'# V0.73 16.02.1999 TB  Allow comment in figure, and 0 = empty cell, small
cleanup
'# V0.81 17.02.1999 TB  Added V0.72 Solver, Corrected Batch
'# V0.82 18.02.1999 CMF Corrected solver array initialization
'# V0.83 19.02.1999 TB  Modularized Solve routines
'# V0.84 25.02.1999 TB  Implemented X,Y symmetry, Changed from OnKey to polled,
'#                      Allow Batch save of graphics
'# V0.85 01.03.1999 TB  Minor corrections, Corrected a wrong filename in batch
'# V0.86 02.03.1999 TB  Minor corrections, Revised batch to NOT clear screen each
loop
'# V0.90 19.03.1999 TB  Corrections, Cube count in drawing,
'#                      Save to TXT files, append if same name.
'#                      Corrected fig nr in save
'#                      The endmark /SOMAEND is no more needed
'# V0.9B 28.03.1999 CMF More speed on Answer: routine
'# V0.91 15.04.1999 TB  Detect the prescence of an HTML header and start
'#                        listing header after <!/SOMAHEAD> identifier
'#                      Make piece count color depend on possibility of solve
'#                      Clear grafic label before new label, limit label to 1
line.
'#                      Clear G() below C190
'# V1.00 03.05.1999 CMF Allow partial solve
'# V1.01 07.06.1999 CMF Slightly speed improved
'# V1.09 05.08.1999 TB  select 30 or 45 deg viewing angle
'#                      allow the finding of more solutions to a figure.
'# V1.10 09.08.1999 TB  Changed the keys for rotation
'# V1.10 02.08.1999 TB  Changed the keys for rotation
'# V1.10x 02.09.1999 TB Accept Sivy's VLTZABP piece notation
'# V1.20 17.09.1999 CMF Small array cleaning
'# V1.21  7.01.2001 TB  Support graphic show of SOMA+Plus figures @Rlop1
'#
'####################################################################


'####################################################################
'#
'# Initializations
'#
'####################################################################



   DEFINT A-Z             'Integer for faster compute
   OPTION BASE 0          'important! The "A" lattice in the
                          'ISLAND.BAS structure needs the
                          'A(0,0,0) coordinate.
   DIM A(17, 17, 17)      'ISLAND is needing an extra layer
                          'around using indices 0 and 17 around
                          'the 16*16*16 box
   DIM S(7, 160, 4)       'S1-S7 CUB data
   DIM E0(27, 6), E1(27) 'ISLAND matrix, temp ISLAND data
   DIM F(27), G(27)       'temp CUB data, PARITY string
   DIM AA(7, 4), B(27)    'final answer, intersection check
   DIM C(4), D(7), E(60) 'pointers
   DIM Fld(17, 17, 17) AS INTEGER 'for drawing
   DIM SymUse(27)         'Cube in use if 1
```

```
   SCREEN 12: 'cls
   WINDOW (-519, -299)-(120, 180)  ' maxx-639,maxy-479,maxx,maxy
   NF = 1
   Vue = 0             'Initial View angle
   GOSUB NewFil        'Initially start by selecting a file


'####################################################################
'#
'# Program user interface
'#
'####################################################################

ProgStart:  IK$ = INKEY$

            IF IK$ = "" THEN GOTO ProgStart
            IK = ASC(IK$)
            IF IK = 27 THEN
              GOSUB Clrs
              LOCATE 5, 1
              INPUT "Quit SOLVE (Y/N) ?  ", A$
              IF A$ = "Y" OR A$ = "y" THEN CLS : END
              GOSUB Clrs
              IK = 1
            END IF
            IF IK <> 0 THEN GOTO ProgStart
            IK$ = MID$(IK$, 2)
            IK = ASC(IK$)
            IF IK = 59 THEN NF = 1: GOSUB NewFig       'Key1 New figure
            IF IK = 60 THEN GOSUB Solvefig             'Key2 Solve figure
            IF IK = 61 THEN GOSUB Showfigur            'Key3 Show figure
            IF IK = 62 THEN GOSUB PrintFig             'Key4 Show solution
            IF IK = 63 THEN NF = 1: GOSUB EnterFig     'Key5 Input figure
            IF IK = 64 THEN NF = 1: GOSUB Batch        'Key6 Batch
            IF IK = 65 THEN NF = 1: GOSUB NextFig      'Key7 Next figure
            IF IK = 66 THEN GOSUB SaveBMP              'Key8 Save BMP image
            IF IK = 67 THEN GOSUB SaveFigur            'Key9 Save figure
            IF IK = 68 THEN NF = 1: GOSUB NewFil       'Key10 New File & Exit
            IF IK = 79 THEN GOSUB RotateR              'End=(Right rotate)
            IF IK = 77 THEN GOSUB RotateR              'Right
            IF IK = 71 THEN GOSUB RotateL              'Home=(Left rotate)
            IF IK = 75 THEN GOSUB RotateL              'Left
            IF IK = 72 THEN Vue = 1: GOSUB ShowGraphic 'Up
            IF IK = 80 THEN Vue = 0: GOSUB ShowGraphic 'Down

          GOTO ProgStart           'Wait for function keys

'======================
'Select the file
NewFil:  GOSUB NextFile  'Ask for the filename
         ON ERROR GOTO 0
         GOSUB Clrs
         COLOR 10: LOCATE 10, 1: PRINT "NOW: select a NEW FIGURE number:"
 LOCATE 27, 1 'New figure +--2---++--3---++--4---++--5---++--6---++--7---+
 COLOR 10: PRINT " ______                                                  ";

         GOSUB NewFig   'Force the question for a New figure like [F1]

         RETURN

'======================
'Select the figure number
NewFig:  GOSUB Clrs
         GOSUB FileHead  'Open the file and print the Header
           PRINT
ClrKbd1:  IF INKEY$ <> "" THEN GOTO ClrKbd1
           COLOR 11: PRINT "Enter Figure number: ";
           COLOR 3: PRINT " ";
           COLOR 11: INPUT "", Xx$
```

```
          Xx$ = UCASE$(Xx$)
          IF Xx$ = "" THEN RETURN
        GOSUB Clrs
        GOSUB ScanFile  'Scan the file to find the Figure, and load the data
          IF Aeof = 1 THEN
            GOSUB Clrs: COLOR 12: LOCATE 10, 1
            PRINT "The figure number " + Xx$ + " is NOT found."
            RETURN
          END IF

        NoSol = 1        'Dont show solution YET
        GOSUB ShowGraphic  'Now show the structure (With true numbers)

      LOCATE 20, 53: PRINT SPACE$(27);
      COLOR 14
      LOCATE 20, 53: PRINT "SOMA"; Xx$; " in "; file$; ".DAT"

 LOCATE 27, 1 'New solve  +--2---++--3---++--4---++--5---++--6---++--7---+
 COLOR 10: PRINT "         ______                                          ";
      RETURN

'======================
'Next figure
NextFig: GOSUB Clrs
        T$ = Xx$: Xx$ = ""
        GOSUB ReadLin   'Scan the file to find the next '/SOMA'
                        'Xx$ is reassigned
          IF Aeof = 1 THEN
            Aeof = 2
            GOSUB Clrs: COLOR 12: LOCATE 10, 1
            PRINT "End Of File."
            PRINT "NO Next figure is found."
            CLOSE #1 'Close all files
            OPEN file$ + ".HTM" FOR INPUT AS #1  'Restart file
            RETURN
          END IF

        NoSol = 1        'Dont show solution YET
        GOSUB ShowGraphic  'Now show the structure (With true numbers)

      LOCATE 20, 53: PRINT SPACE$(27);
      COLOR 14
      LOCATE 20, 53: PRINT "SOMA"; Xx$; " in "; file$; ".DAT"

 LOCATE 27, 1 'New solve  +--2---++--3---++--4---++--5---++--6---++--7---+
 COLOR 10: PRINT "         ______          ______                  ______ ";
      RETURN

'======================
'Solve the figure
Solvefig: IF FigX = 0 OR FigY = 0 OR FigZ = 0 THEN
            GOSUB Clrs
            COLOR 12: LOCATE 10, 1: PRINT "Load a 'New figure' before solving."
            RETURN
          END IF

        GOSUB Clrs

IF NF = 1 OR Vink <> 0 THEN    'New figure OR Angle has changed
        Reca = 1
        Vink = 0
        GOSUB PreSolve
        GOSUB FindSolve
        NF = 0                 'Allow multiple solves from now on

ELSE                           'Multiple solves
        Reca = Reca + 1
        GOSUB Nxtsolv
END IF
        IF solve = 1 THEN
           GOSUB PostSolve     'for output and from A()->Fld()

           IF NoSol = 0 THEN GOSUB ShowGraphic'Now show the graphic
```

```
        END IF

 LOCATE 27, 1 'New color  +--2---++--3---++--4---++--5---++--6---++--7---+
 COLOR 10: PRINT "                  ______                                 ";
        RETURN

'======================
'Show the resulting figure
Showfigur: NoSol = (NoSol + 1) AND 1
           GOSUB ShowGraphic 'Now show the graphic
 LOCATE 27, 1 'New txt    +--2---++--3---++--4---++--5---++--6---++--7---+
 COLOR 10: PRINT "                          ______                         ";
           RETURN

'======================
'Save the resulting figure
SaveFigur: GOSUB SaveResult
           GOSUB Clrs
           RETURN

'======================
Batch:   GOSUB NextFile  'Leave File$+".HTM" open as #2
         ON ERROR GOTO 0
         GOSUB Clrs

         BatOn = 1

           OPEN file$ + ".HTB" FOR OUTPUT AS #3 'Make a new file
           PRINT #3, " | BATCH Solutions to SOMA figures from "; file$; ".HTM"
           PRINT #3, " | Made by 'Bundgård/McFarren's Solution program."
           PRINT #3, " +------"

'Figure Size
' sizexx = 0: sizeyy = 0
BatchLop:
        GOSUB Clrs

        IF INKEY$ <> "" THEN
          LOCATE 5, 1
          INPUT "Quit Batch (Y/N) ?  ", A$
           IF A$ = "Y" OR A$ = "y" THEN GOTO BatchE
          GOSUB Clrs
        END IF

BatchL:
      IF EOF(1) THEN GOTO BatchE
      LINE INPUT #1, A$
      PRINT #3, A$
      IF INSTR(A$, "/SOMAEND") <> 0 THEN
BatchE: CLOSE #3
        COLOR 10: LOCATE 10, 1: PRINT "Batch Solving ended: "; file$; ".HTB is
ready."
        BatOn = 0
        RETURN  '=====>
      END IF
      IF INSTR(A$, "/SOMA") = 0 THEN GOTO BatchL
      Here$ = MID$(A$, 1, 8)

      GOSUB FoundFig
      GOSUB ShowGraphic     'Now show the structure (With true numbers)

''Figure Size
' PRINT pxm - pxu + 15, pym - pyu + 17
' pxs = pxm - pxu + 15: pys = pym - pyu + 17
' IF pxs > sizexx THEN sizexx = pxs
' IF pys > sizeyy THEN sizeyy = pys
' PRINT sizexx, sizeyy

      LOCATE 20, 53: PRINT SPACE$(27);
      COLOR 14
      LOCATE 20, 53: PRINT "SOMA"; Xx$; " in "; file$; ".DAT"
      GOSUB PreSolve
```

```
      GOSUB FindSolve
         GOSUB PostSolve    'for output and from A()->Fld()
         NoSol = 0          'Allow solution if any
         GOSUB ShowGraphic  'Now show the graphic
'Save the resulting figure
      IF solve = 0 THEN
        PRINT #3, "; No Solution !!"
      END IF

         FOR J = 1 TO FigY
           FOR K = FigZ TO 1 STEP -1
             PRINT #3, "/";
             FOR I = 1 TO FigX
               IF Fld(I, J, K) = 0 THEN
                 PRINT #3, ".";
               ELSE
                 IF Fld(I, J, K) > 9 THEN Fld(I, J, K) = 9
                 PRINT #3, USING "#"; Fld(I, J, K);
               END IF
             NEXT I
           NEXT K
           PRINT #3, ""
         NEXT J
         PRINT #3, ""

     GOTO BatchLop

'######################################################################
'#
'# Make a nice screen
'#
'######################################################################
Clrs:
     FOR Ly = 1 TO 26
     LOCATE Ly, 1: PRINT SPACE$(50);
     NEXT Ly
   COLOR 14
     LOCATE 1, 53: PRINT "SOMA solution finder V1.20"
     LOCATE 2, 53: PRINT "Bundgård / McFarren 1999"
   COLOR 11
     LOCATE 26, 1: PRINT SPACE$(63);
     LOCATE 26, 58: PRINT "  Tilt:   [Up] [Down]"
     LOCATE 27, 1: PRINT SPACE$(63);
     LOCATE 27, 58: PRINT "Rotate: [Left] [Right]"
   COLOR 3
     LOCATE 28, 1
     PRINT "+--1---++--2---++--3---++--4---++--5---+";
     PRINT "+--6---++--7---++--8---++--9---++--10--+";
   COLOR 11
     LOCATE 29, 1
           '--------========--------========--------'
     PRINT " New             Solved  Solved  Input  ";
     PRINT " Batch   Next    Save    Save   New File";
     LOCATE 30, 1
     PRINT " Figure  Solve   Color   Text    Figure ";
     PRINT " Solve   Figure  BMP     Figure  & Exit ";
           '--------========--------========--------'
   COLOR 15
     LOCATE 1, 1
     RETURN

'######################################################################
'#
'# Print the SOMA figure numerically
'#
'######################################################################
PrintFig:
     GOSUB Clrs
     A = FigZ * (FigY + 1)
     IF A < 25 THEN

       PRINT "This is the Figure in text."
         FOR K = FigZ TO 1 STEP -1
```

```
          FOR J = 1 TO FigY
            FOR I = 1 TO FigX
              PRINT USING "#"; Fld(I, J, K);
            NEXT I
            PRINT
          NEXT J
          PRINT
        NEXT K
        RETURN

     ELSE

       A = INT(25 / (FigY + 1))
       K = FigZ

PrintPart:
       COLOR 15
       GOSUB Clrs
       PRINT "This is the Figure in text."
       FOR N = 1 TO A
         FOR J = 1 TO FigY
           IF J = 1 THEN
             PRINT USING "Z = ##  "; K;
           ELSE
             PRINT "        ";
           END IF
           FOR I = 1 TO FigX
             PRINT USING "#"; Fld(I, J, K);
           NEXT I
           PRINT
         NEXT J
         PRINT
         K = K - 1
         IF K = 0 THEN RETURN
       NEXT N
ClrKbd4:  IF INKEY$ <> "" THEN GOTO ClrKbd4 'Clear key buffer
       COLOR 11: INPUT "[Enter] for next, [Q][Enter] to Quit."; A$
       IF A$ = "Q" OR A$ = "q" THEN RETURN
       GOTO PrintPart

     END IF
     RETURN

'####################################################################
'#
'# Display the SOMA figure graphically
'#    RotateL:     Rotate left and show figure
'#    RotateR:     Rotate right and show figure
'#    ShowGraphic: Draw the figure
'#
'####################################################################
'Rotate left and show figure
RotateL: 'Left
  Vink = Vink + 1 AND 3
  FOR Zz = 1 TO 16                                          ' A > B
    FOR Yy = 1 TO 8                                         ' ^   v
      FOR Xx = 1 TO 8                                       ' D < C
           A = Fld(Xx, Yy, Zz)
           Fld(Xx, Yy, Zz) = Fld(Yy, 17 - Xx, Zz)           'A<-D
           Fld(Yy, 17 - Xx, Zz) = Fld(17 - Xx, 17 - Yy, Zz) 'D<-C
           Fld(17 - Xx, 17 - Yy, Zz) = Fld(17 - Yy, Xx, Zz) 'C<-B
           Fld(17 - Yy, Xx, Zz) = A                         'B<-A
      NEXT Xx
    NEXT Yy
  NEXT Zz
  GOTO MoveBack  'Move structure back to 0,0,0

'====================================================================
'Rotate right and show figure
RotateR: 'Right
  Vink = Vink - 1 AND 3
  FOR Zz = 1 TO 16                                          ' A < B
    FOR Yy = 1 TO 8                                         ' v   ^
```

```
      FOR Xx = 1 TO 8                                          ' D > C
           A = Fld(Xx, Yy, Zz)
           Fld(Xx, Yy, Zz) = Fld(17 - Yy, Xx, Zz)              'A<-B
           Fld(17 - Yy, Xx, Zz) = Fld(17 - Xx, 17 - Yy, Zz) 'B<-C
           Fld(17 - Xx, 17 - Yy, Zz) = Fld(Yy, 17 - Xx, Zz) 'C<-D
           Fld(Yy, 17 - Xx, Zz) = A                            'D<-A
      NEXT Xx
    NEXT Yy
  NEXT Zz

'Move structure back to 0,0,0
MoveBack:
  Xxmin = 16: Xxmax = 0: Yymin = 16: Yymax = 0: Zzmin = 16: Zzmax = 0
  FOR Zz = 1 TO 16
    FOR Yy = 1 TO 16
      FOR Xx = 1 TO 16
        IF Fld(Xx, Yy, Zz) <> 0 THEN
        IF Xx < Xxmin THEN Xxmin = Xx
        IF Xx > Xxmax THEN Xxmax = Xx
        IF Yy < Yymin THEN Yymin = Yy
        IF Yy > Yymax THEN Yymax = Yy
        IF Zz < Zzmin THEN Zzmin = Zz
        IF Zz > Zzmax THEN Zzmax = Zz
        END IF
      NEXT Xx
    NEXT Yy
  NEXT Zz

'Borrow the A array for the transposition
  FOR Zz = 1 TO 16
    FOR Yy = 1 TO 16
      FOR Xx = 1 TO 16
        A(Xx, Yy, Zz) = 0
      NEXT Xx
    NEXT Yy
  NEXT Zz

  FOR Zz = Zzmin TO Zzmax
    FOR Yy = Yymin TO Yymax
      FOR Xx = Xxmin TO Xxmax
        A(Xx - Xxmin + 1, Yy - Yymin + 1, Zz - Zzmin + 1) = Fld(Xx, Yy, Zz)
      NEXT Xx
    NEXT Yy
  NEXT Zz

  FOR Zz = 1 TO 16
    FOR Yy = 1 TO 16
      FOR Xx = 1 TO 16
        Fld(Xx, Yy, Zz) = A(Xx, Yy, Zz)
      NEXT Xx
    NEXT Yy
  NEXT Zz


  FigX = Xxmax - Xxmin + 1
  FigY = Yymax - Yymin + 1
  FigZ = Zzmax - Zzmin + 1

  GOTO ShowGraphic

'================================================================
'   z|
'    |_____X
'   /
'  y
ShowGraphic:
'45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45
  IF Vue = 0 THEN  '45 deg View
'    Xmax = Xx + 1
'    Ymax = Yy + 1
'    Zmax = 16 - Zz
'    Xsize = Xmax * 9 + Ymax * 9 + 9
'    Ysize = Zmax * 11 + Xmax * 3 + Ymax * 3 + 7
```

```
                'ZW=8 YW=4 XW=7
    minx = -105  '  0 * 7 - 15 * 7          ' Xmin * XW - Ymax * XW
    maxx = 120   ' 15 * 7 - 0 * 7 + (2*7+1) ' Xmax * XW - Ymin * XW +
Figwidth(2*XW+1)
    miny = -120  '-15 * 4 - 15 * 4 + 0 * 8  '-Ymax * YW - Xmax * YW + Zmin * ZW
    maxy = 137   ' -0 * 4 - 0 * 4 + 15 * 8 +(8+2*4+1) ' -Ymin * YW - Xmin * YW +
Zmax * ZW +Figheight(ZW+2*YW+1)

'For check of area limits
'Fld(0, 0, 0) = 4: Fld(15, 0, 0) = 4: Fld(0, 15, 0) = 4: Fld(15, 15, 0) = 4
'Fld(0, 0, 15) = 4: Fld(15, 0, 15) = 4: Fld(0, 15, 15) = 4: Fld(15, 15, 15) = 4


'Make a white area
   FOR N = miny TO maxy
   LINE (minx, N)-(maxx, N), 15
   NEXT N
'Make Yellow border
   LINE (minx - 1, miny - 1)-(maxx + 1, miny - 1), 14
   LINE (minx - 1, maxy + 1)-(maxx + 1, maxy + 1), 14
   LINE (minx - 1, miny - 1)-(minx - 1, maxy + 1), 14
   LINE (maxx + 1, miny - 1)-(maxx + 1, maxy + 1), 14

''Figure Size (Find real size of the figure in view)
' pxm = -200: pym = -200
' pxu = 200: pyu = 200

  FOR Zz = 0 TO 15
    FOR Yy = 0 TO 15
      FOR Xx = 0 TO 15
        Fig = Fld(Xx + 1, Yy + 1, Zz + 1)
        IF Fig <> 0 THEN
           px = 0 + Xx * 7 - Yy * 7        'px = 6 + Xx * XW - Yy * XW
           py = -Yy * 4 - Xx * 4 + Zz * 8  'py = -Yy * YW - Xx * YW + Zz * ZW

''Figure Size
'           IF px > pxm THEN pxm = px
'           IF py > pym THEN pym = py
'           IF px < pxu THEN pxu = px
'           IF py < pyu THEN pyu = py

          GOSUB Cube
        END IF
      NEXT Xx
    NEXT Yy
  NEXT Zz

  RETURN

' +-----+ XW=7
'        s
'      ss1ss
'    ss11111ss
'  ss111111111ss
' s1111111111111s  +
' sss111111111sss  ¦YW=4
' s22ss11111ss33s  ¦
' s2222ss1ss3333s  +
' s222222s333333s  +
' s222222s333333s  ¦
' s222222s333333s  ¦
' s222222s333333s  ¦ZW=8
' s222222s333333s  ¦
'  ss2222s3333ss   ¦
'    ss22s33ss     ¦
'      sssss       +
'        s

Cube:
  IF Fig = 9 THEN c1 = 14: c2 = 12: c3 = 4 'Yel,Red,Dred
  IF Fig = 1 THEN c1 = 13: c2 = 13: c3 = 5 'Purple
  IF Fig = 2 THEN c1 = 12: c2 = 12: c3 = 4 'Red
```

```
  IF Fig = 3 THEN c1 = 14: c2 = 14: c3 = 6 'Yel
  IF Fig = 4 THEN c1 = 9: c2 = 9: c3 = 1 'Blu
  IF Fig = 5 THEN c1 = 10: c2 = 10: c3 = 2 'Gren
  IF Fig = 6 THEN c1 = 7: c2 = 7: c3 = 8 'Gray
  IF Fig = 7 THEN c1 = 11: c2 = 11: c3 = 3 'Cyan
  IF NoSol = 1 THEN c1 = 14: c2 = 12: c3 = 4 'Yel,Red,Dred

Sx = px
Sy = py + 16: PSET (Sx + 7, Sy), 0                    '.......s.......
Sy = py + 15: LINE (Sx + 5, Sy)-(Sx + 6, Sy), 0       '.....ss1ss.....
              PSET (Sx + 7, Sy), c1
              LINE (Sx + 8, Sy)-(Sx + 9, Sy), 0
Sy = py + 14: LINE (Sx + 3, Sy)-(Sx + 4, Sy), 0       '...ss11111ss...
              LINE (Sx + 5, Sy)-(Sx + 9, Sy), c1
              LINE (Sx + 10, Sy)-(Sx + 11, Sy), 0
Sy = py + 13: LINE (Sx + 1, Sy)-(Sx + 2, Sy), 0       '.ss111111111ss.
              LINE (Sx + 3, Sy)-(Sx + 11, Sy), c1
              LINE (Sx + 12, Sy)-(Sx + 13, Sy), 0
Sy = py + 12: PSET (Sx, Sy), 0                        's1111111111111s
              LINE (Sx + 1, Sy)-(Sx + 13, Sy), c1
              PSET (Sx + 14, Sy), 0
Sy = py + 11: LINE (Sx + 0, Sy)-(Sx + 2, Sy), 0       'sss111111111sss
              LINE (Sx + 3, Sy)-(Sx + 11, Sy), c1
              LINE (Sx + 12, Sy)-(Sx + 14, Sy), 0
Sy = py + 10: PSET (Sx, Sy), 0                        's22ss11111ss33s
              LINE (Sx + 1, Sy)-(Sx + 2, Sy), c2
              LINE (Sx + 3, Sy)-(Sx + 4, Sy), 0
              LINE (Sx + 5, Sy)-(Sx + 9, Sy), c1
              LINE (Sx + 10, Sy)-(Sx + 11, Sy), 0
              LINE (Sx + 12, Sy)-(Sx + 13, Sy), c3
              PSET (Sx + 14, Sy), 0
Sy = py + 9: PSET (Sx, Sy), 0                         's2222ss1ss3333s
             LINE (Sx + 1, Sy)-(Sx + 4, Sy), c2
             LINE (Sx + 5, Sy)-(Sx + 6, Sy), 0
             PSET (Sx + 7, Sy), c1
             LINE (Sx + 8, Sy)-(Sx + 9, Sy), 0
             LINE (Sx + 10, Sy)-(Sx + 13, Sy), c3
             PSET (Sx + 14, Sy), 0
Sy = py + 8: PSET (Sx, Sy), 0                         's222222s333333s
             LINE (Sx + 1, Sy)-(Sx + 6, Sy), c2
             PSET (Sx + 7, Sy), 0
             LINE (Sx + 8, Sy)-(Sx + 13, Sy), c3
             PSET (Sx + 14, Sy), 0
Sy = py + 7: PSET (Sx, Sy), 0                         's222222s333333s
             LINE (Sx + 1, Sy)-(Sx + 6, Sy), c2
             PSET (Sx + 7, Sy), 0
             LINE (Sx + 8, Sy)-(Sx + 13, Sy), c3
             PSET (Sx + 14, Sy), 0
Sy = py + 6: PSET (Sx, Sy), 0                         's222222s333333s
             LINE (Sx + 1, Sy)-(Sx + 6, Sy), c2
             PSET (Sx + 7, Sy), 0
             LINE (Sx + 8, Sy)-(Sx + 13, Sy), c3
             PSET (Sx + 14, Sy), 0
Sy = py + 5: PSET (Sx, Sy), 0                         's222222s333333s
             LINE (Sx + 1, Sy)-(Sx + 6, Sy), c2
             PSET (Sx + 7, Sy), 0
             LINE (Sx + 8, Sy)-(Sx + 13, Sy), c3
             PSET (Sx + 14, Sy), 0
Sy = py + 4: PSET (Sx, Sy), 0                         's222222s333333s
             LINE (Sx + 1, Sy)-(Sx + 6, Sy), c2
             PSET (Sx + 7, Sy), 0
             LINE (Sx + 8, Sy)-(Sx + 13, Sy), c3
             PSET (Sx + 14, Sy), 0
Sy = py + 3: LINE (Sx + 1, Sy)-(Sx + 2, Sy), 0        '.ss2222s3333ss.
             LINE (Sx + 3, Sy)-(Sx + 6, Sy), c2
             PSET (Sx + 7, Sy), 0
             LINE (Sx + 8, Sy)-(Sx + 11, Sy), c3
             LINE (Sx + 12, Sy)-(Sx + 13, Sy), 0
Sy = py + 2: LINE (Sx + 3, Sy)-(Sx + 4, Sy), 0        '...ss22s33ss...
             LINE (Sx + 5, Sy)-(Sx + 6, Sy), c2
             PSET (Sx + 7, Sy), 0
             LINE (Sx + 8, Sy)-(Sx + 9, Sy), c3
             LINE (Sx + 10, Sy)-(Sx + 11, Sy), 0
```

```
Sy = py + 1: LINE (Sx + 5, Sy)-(Sx + 9, Sy), 0       '.....sssss.....
Sy = py:     PSET (Sx + 7, Sy), 0                    '.......s.......

'30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
ELSE  '30 deg View

'ZW=8 YW=3 XW=7
    minx = -105  '  0 * 7 - 15 * 7          ' Xmin * XW - Ymax * XW
    maxx = 120   ' 15 * 7 - 0 * 7 + (2*7+1) ' Xmax * XW - Ymin * XW +
Figwidth(2*XW+1)
    miny = -90   '-15 * 3 - 15 * 3 + 0 * 8  '-Ymax * YW - Xmax * YW + Zmin * ZW
    maxy = 135   ' -0 * 3 - 0 * 3 + 15 * 8 +(8+2*3+1) ' -Ymin * YW - Xmin * YW +
Zmax * ZW +Figheight(ZW+2*YW+1)

'For the sake of the save algorithm we use the larger 45 deg values
    minx = -105
    maxx = 120
    miny = -120
    maxy = 137


'Make a white area
   FOR N = miny TO maxy
   LINE (minx, N)-(maxx, N), 15
   NEXT N
'Make Yellow border
   LINE (minx - 1, miny - 1)-(maxx + 1, miny - 1), 14
   LINE (minx - 1, maxy + 1)-(maxx + 1, maxy + 1), 14
   LINE (minx - 1, miny - 1)-(minx - 1, maxy + 1), 14
   LINE (maxx + 1, miny - 1)-(maxx + 1, maxy + 1), 14

  FOR Zz = 0 TO 15
    FOR Yy = 0 TO 15
      FOR Xx = 0 TO 15
        Fig = Fld(Xx + 1, Yy + 1, Zz + 1)
        IF Fig <> 0 THEN
           px = 0 + Xx * 7 - Yy * 7        'px = 6 + Xx * XW - Yy * XW
           py = -Yy * 3 - Xx * 3 + Zz * 8  'py = -Yy * YW - Xx * YW + Zz * ZW
         'At 30 deg YW=3 instead of 4
           GOSUB Cube30
        END IF
      NEXT Xx
    NEXT Yy
  NEXT Zz

  RETURN

' +-----+ XW=7
'       s
'     sss1sss
'  sss1111111sss
' s1111111111111s  +
' ssss1111111ssss  ¦YW=3
' s222sss1sss333s  +
' s222222s333333s  +
' s222222s333333s  ¦
' s222222s333333s  ¦
' s222222s333333s  ¦ZW=8
' s222222s333333s  ¦
' s222222s333333s  ¦
'  sss222s333sss   ¦
'     sssssss      +
'       s

Cube30:
  IF Fig = 9 THEN c1 = 14: c2 = 12: c3 = 4 'Yel,Red,Dred
  IF Fig = 1 THEN c1 = 13: c2 = 13: c3 = 5 'Purple
  IF Fig = 2 THEN c1 = 12: c2 = 12: c3 = 4 'Red
  IF Fig = 3 THEN c1 = 14: c2 = 14: c3 = 6 'Yel
  IF Fig = 4 THEN c1 = 9: c2 = 9: c3 = 1 'Blu
  IF Fig = 5 THEN c1 = 10: c2 = 10: c3 = 2 'Gren
  IF Fig = 6 THEN c1 = 7: c2 = 7: c3 = 8 'Gray
  IF Fig = 7 THEN c1 = 11: c2 = 11: c3 = 3 'Cyan
```

```
  IF NoSol = 1 THEN c1 = 14: c2 = 12: c3 = 4 'Yel,Red,Dred

Sx = px
Sy = py + 14: PSET (Sx + 7, Sy), 0                    '.......s.......
Sy = py + 13: LINE (Sx + 4, Sy)-(Sx + 6, Sy), 0       '....sss1sss....
              PSET (Sx + 7, Sy), c1
              LINE (Sx + 8, Sy)-(Sx + 10, Sy), 0
Sy = py + 12: LINE (Sx + 1, Sy)-(Sx + 3, Sy), 0       '.sss1111111sss.
              LINE (Sx + 4, Sy)-(Sx + 10, Sy), c1
              LINE (Sx + 11, Sy)-(Sx + 13, Sy), 0
Sy = py + 11: PSET (Sx, Sy), 0                        's1111111111111s
              LINE (Sx + 1, Sy)-(Sx + 13, Sy), c1
              PSET (Sx + 14, Sy), 0
Sy = py + 10: LINE (Sx + 0, Sy)-(Sx + 3, Sy), 0       'ssss1111111ssss
              LINE (Sx + 4, Sy)-(Sx + 10, Sy), c1
              LINE (Sx + 11, Sy)-(Sx + 14, Sy), 0
Sy = py + 9: PSET (Sx, Sy), 0                         's222sss1sss333s
             LINE (Sx + 1, Sy)-(Sx + 3, Sy), c2
             LINE (Sx + 4, Sy)-(Sx + 6, Sy), 0
             PSET (Sx + 7, Sy), c1
             LINE (Sx + 8, Sy)-(Sx + 10, Sy), 0
             LINE (Sx + 11, Sy)-(Sx + 13, Sy), c3
             PSET (Sx + 14, Sy), 0
Sy = py + 8: PSET (Sx, Sy), 0                         's222222s333333s
             LINE (Sx + 1, Sy)-(Sx + 6, Sy), c2
             PSET (Sx + 7, Sy), 0
             LINE (Sx + 8, Sy)-(Sx + 13, Sy), c3
             PSET (Sx + 14, Sy), 0
Sy = py + 7: PSET (Sx, Sy), 0                         's222222s333333s
             LINE (Sx + 1, Sy)-(Sx + 6, Sy), c2
             PSET (Sx + 7, Sy), 0
             LINE (Sx + 8, Sy)-(Sx + 13, Sy), c3
             PSET (Sx + 14, Sy), 0
Sy = py + 6: PSET (Sx, Sy), 0                         's222222s333333s
             LINE (Sx + 1, Sy)-(Sx + 6, Sy), c2
             PSET (Sx + 7, Sy), 0
             LINE (Sx + 8, Sy)-(Sx + 13, Sy), c3
             PSET (Sx + 14, Sy), 0
Sy = py + 5: PSET (Sx, Sy), 0                         's222222s333333s
             LINE (Sx + 1, Sy)-(Sx + 6, Sy), c2
             PSET (Sx + 7, Sy), 0
             LINE (Sx + 8, Sy)-(Sx + 13, Sy), c3
             PSET (Sx + 14, Sy), 0
Sy = py + 4: PSET (Sx, Sy), 0                         's222222s333333s
             LINE (Sx + 1, Sy)-(Sx + 6, Sy), c2
             PSET (Sx + 7, Sy), 0
             LINE (Sx + 8, Sy)-(Sx + 13, Sy), c3
             PSET (Sx + 14, Sy), 0
Sy = py + 3: PSET (Sx, Sy), 0                         's222222s333333s
             LINE (Sx + 1, Sy)-(Sx + 6, Sy), c2
             PSET (Sx + 7, Sy), 0
             LINE (Sx + 8, Sy)-(Sx + 13, Sy), c3
             PSET (Sx + 14, Sy), 0
Sy = py + 2: LINE (Sx + 1, Sy)-(Sx + 3, Sy), 0        '.sss222s333sss.
             LINE (Sx + 4, Sy)-(Sx + 6, Sy), c2
             PSET (Sx + 7, Sy), 0
             LINE (Sx + 8, Sy)-(Sx + 10, Sy), c3
             LINE (Sx + 11, Sy)-(Sx + 13, Sy), 0
Sy = py + 1: LINE (Sx + 4, Sy)-(Sx + 10, Sy), 0       '....sssssss....
Sy = py:     PSET (Sx + 7, Sy), 0                     '.......s.......

END IF

             RETURN

'####################################################################
'#
'# Manual input
'#
'####################################################################
'Enter the figure
EnterFig:
    CLOSE  'Close all files
```

```
    GOSUB Clrs
ClrKbd3:  IF INKEY$ <> "" THEN GOTO ClrKbd3
    COLOR 15: PRINT "Manual Figure Entry."
    Zi = 16: Xi = 16: Yi = 16
    INPUT "Erase figure: [Y]es / [N]o: ", A$
    IF A$ = "Y" OR A$ = "y" THEN
      FOR K = 0 TO 17      'Clear the array
        FOR J = 0 TO 17
          FOR I = 0 TO 17
            Fld(I, J, K) = 0
          NEXT I
        NEXT J
      NEXT K
      GOSUB ShowGraphic  'Now show the structure (With true numbers)
    END IF

'Disable key trap for now
    KEY(12) OFF: KEY(13) OFF   'Left, Right

    GOSUB Clrs
    COLOR 14
    LOCATE 1, 1: PRINT "[PgUp]/[PgDn]=Level Up/Dn [Esc]=StopEditing"
                 PRINT "[Up][Dn][Left][Right] Move [Ins]=Set/Res"
                 PRINT "[Del]=Clear  [Home]/[End]=Rotates"

    Zp = 1: Xp = 1: Yp = 1
ReDrw:
      LOCATE 4, 1: COLOR 12: PRINT "Level "; Zp
        FOR A = 1 TO Zi
          LOCATE 4 + A, 1: COLOR 12: PRINT "¦  ";
        NEXT A
        LOCATE 5 + Zi - Zp, 1: COLOR 14: PRINT "¦<-";
      COLOR 2
      FOR Ypi = 1 TO Yi
        FOR Xpi = 1 TO Xi
          LOCATE 5 + Ypi, 10 + Xpi
          IF Fld(Xpi, Ypi, Zp) <> 0 THEN PRINT "*";  ELSE PRINT ".";
        NEXT Xpi
      NEXT Ypi
      LOCATE 5 + Yp, 10 + Xp: COLOR 15
      IF Fld(Xp, Yp, Zp) <> 0 THEN PRINT "*";  ELSE PRINT ".";
      COLOR 14

    Cube = 0
    FOR K = 0 TO 17      'Count the array
      FOR J = 0 TO 17
        FOR I = 0 TO 17
          IF Fld(I, J, K) THEN Cube = Cube + 1
        NEXT I
      NEXT J
    NEXT K
    LOCATE 22, 13

'Make piece count color depend on possibility
'Green: 3,4,7,8,11,12,15,16,19,20,23,24,27
'Red:   1,2,5,6,9,10,13,14,17,18,21,22,25,26
'(piece+1) mod 4 = 0,1 Green.  2,3 Red
    COLOR 12 'Red=No Solve
    A = (Cube + 1) MOD 4
    IF A = 0 OR A = 1 THEN COLOR 10 'Green Maybe
    IF Cube = 0 THEN COLOR 12 'No cubes at all
    PRINT Cube; " Cubes"


InKy:
   A$ = INKEY$
   IF A$ = "" THEN GOTO InKy
   A = ASC(A$)
   IF A = 27 THEN GOTO InpEnd
   IF A <> 0 THEN GOTO InKy

' Home 00,71  Up 00,72    PgUp 00,73   -45
' Left 00,75              Right 00,77  +43
```

```
' End  00,79  Down 00,80  PgDn 00,81
' Ins  00,82  Del 00,83

      A$ = MID$(A$, 2)
      A = ASC(A$)
   IF A = 77 THEN Xp = Xp + 1
   IF A = 75 THEN Xp = Xp - 1
   IF Xp < 1 THEN Xp = 1
   IF Xp > Xi THEN Xp = Xi
   IF A = 80 THEN Yp = Yp + 1
   IF A = 72 THEN Yp = Yp - 1
   IF Yp < 1 THEN Yp = 1
   IF Yp > Yi THEN Yp = Yi
   IF A = 73 THEN Zp = Zp + 1
   IF A = 81 THEN Zp = Zp - 1
   IF Zp < 1 THEN Zp = 1
   IF Zp > Zi THEN Zp = Zi
   IF A = 82 THEN
     IF Fld(Xp, Yp, Zp) = 0 THEN Fld(Xp, Yp, Zp) = 9 ELSE Fld(Xp, Yp, Zp) = 0
   END IF
   IF A = 83 THEN Fld(Xp, Yp, Zp) = 0

   IF A = 79 THEN GOSUB RotateR: GOTO ReDrw
   IF A = 71 THEN GOSUB RotateL: GOTO ReDrw
     GOSUB ShowGraphic  'Now show the structure (With true numbers)
   GOTO ReDrw

InpEnd:
   GOSUB Clrs
   GOSUB MoveBack            'Align to 0,0,0 + GOSUB ShowGraphic
   RETURN

'================================================================
'Get the data for this level
GetLevl:
    PRINT
      LOCATE 4, 1: COLOR 12: PRINT "Level "; Z
        FOR A = 1 TO Zi
          LOCATE 4 + A, 1: COLOR 12: PRINT "¦  ";
        NEXT A
        LOCATE 5 + Zi - Z, 1: COLOR 14: PRINT "¦<-";
      COLOR 2
      FOR Ypi = 1 TO Yi
         LOCATE 5 + Ypi, 10: PRINT STRING$(Xi, ".")
      NEXT Ypi

      FOR Ypi = 1 TO Yi
InLi:   YPf = 0
        FOR A = 1 TO Yi
          COLOR 14: LOCATE 5 + Ypi, 8: PRINT "->";
        NEXT A

        COLOR 10: LOCATE 5 + Ypi, 10
        INPUT "", A$
        IF A$ = "Q" OR A$ = "q" THEN RETURN
        IF A$ = "" THEN
          Ypi = Ypi - 1
          IF Ypi < 1 THEN Ypi = 1
          GOTO InLi
        END IF
        IF YPf = 1 THEN GOTO InLi
        IF LEN(A$) < Xi THEN A$ = A$ + STRING$(Xi, ".")
        IF LEN(A$) > Xi THEN A$ = MID$(A$, 1, Xi)
        FOR Xpi = 1 TO Xi
          V = ASC(MID$(A$, Xpi, 1))
          Fld(Xpi, Ypi, Z) = 0
          IF V >= 48 AND V <= 57 THEN Fld(Xpi, Ypi, Z) = 9: MID$(A$, Xpi, 1) =
"*"
          IF V = 42 THEN Fld(Xpi, Ypi, Z) = 9: MID$(A$, Xpi, 1) = "*"  '*
        NEXT Xpi
        COLOR 14: LOCATE 5 + Ypi, 10: PRINT A$
          GOSUB ShowGraphic  'Now show the structure (With true numbers)
      NEXT Ypi
```

```
      RETURN

'######################################################################
'#
'# File routines
'#
'######################################################################
'Get the file
NextFile:
   A = 0
ClrKbd5:  IF INKEY$ <> "" THEN GOTO ClrKbd5 'Clear key buffer
   CLS
   LOCATE 3, 1
   IF A = 1 THEN COLOR 12: PRINT "File ("; file$; ") NOT found:"
   COLOR 15
   FILES "*.HTM"
   PRINT
   COLOR 3: PRINT "Type [Q][Enter] to Exit."
            PRINT "Type the Filename WITHOUT EXTENSION."
   COLOR 11: PRINT "Filename to use: ";
   COLOR 3: PRINT "SFIGBASE.HTM: ";
   COLOR 11: INPUT " ", file$
   IF file$ = "Q" OR file$ = "q" THEN END
 IF file$ = "" THEN file$ = "SFIGBASE"
   IF INSTR(file$, ".") <> 0 THEN GOTO ClrKbd5
      A$ = ""
   CLOSE  'Close all files
   ON ERROR GOTO NoFile
   A = 0
   OPEN file$ + ".HTM" FOR INPUT AS #1
   IF A = 1 THEN GOTO ClrKbd5
   CLS
   RETURN
NoFile:
   A = 1
   RESUME NEXT

'======================================================================
'Print the header
FileHead:
   GOSUB Clrs
   LOCATE 2, 1
   PRINT A$: A$ = ""   'Message about Not found
   CLOSE #1 'Close all files
   OPEN file$ + ".HTM" FOR INPUT AS #1
   N = 0
   A$ = "Bundgård & McFarren SOMA Solution finder"
Prtop:
   IF LEN(A$) > 50 THEN A$ = MID$(A$, 1, 50)
   PRINT A$
   N = N + 1
PrtSkip:
   LINE INPUT #1, A$

   A = INSTR(A$, "/SOMA")
   IF A > 0 AND A < 40 THEN RETURN
   IF N >= 20 THEN RETURN

   IF INSTR(A$, "<!--") <> 0 THEN GOTO PrtSkip   'Skip HTML comments
   IF INSTR(A$, "<HTML>") <> 0 THEN
HTMLhead:
        LINE INPUT #1, A$
        IF INSTR(A$, "<!/SOMAHEAD>") = 0 AND INSTR(A$, "</HTML>") = 0 THEN GOTO
HTMLhead
        LINE INPUT #1, A$
   END IF

   GOTO Prtop



'======================================================================
'Get the figure
```

```
ScanFile:
    Aeof = 0
    CLOSE #1 'Close all files
    OPEN file$ + ".HTM" FOR INPUT AS #1
ReadLin:
      IF EOF(1) THEN Aeof = 1: RETURN
      LINE INPUT #1, A$
      IF INSTR(A$, "/SOMAEND") <> 0 THEN Aeof = 1: RETURN
      IF INSTR(A$, "/SOMAHEAD") <> 0 THEN GOTO ReadLin
      IF INSTR(A$, "/SOMA" + Xx$) = 0 THEN GOTO ReadLin
      IF INSTR(A$, "/SOMA" + Xx$) < 20 THEN GOTO FoundFig
    GOTO ReadLin

FoundFig:
    L = INSTR(A$, "/SOMA")
    Xx$ = MID$(A$, L + 5, 7)
    L = INSTR(Xx$, " ")
    IF L > 1 THEN Xx$ = MID$(Xx$, 1, L - 1)

'Clear the array
         FOR K = 0 TO 17
           FOR J = 0 TO 17
             FOR I = 0 TO 17
               Fld(I, J, K) = 0
             NEXT I
           NEXT J
         NEXT K

'Get the data
         J = 1: CL = 0
NewCom:  LINE INPUT #1, A$

           P = INSTR(A$, ";")
           IF P > 0 THEN
             IF BatOn THEN PRINT #3, A$
             IF CL < 20 THEN PRINT MID$(A$, P + 1, 50)
             CL = CL + 1
             GOTO NewCom
           END IF

         B$ = A$: Z = 0: P = 0
FindZ:     B$ = MID$(B$, P + 1)
           P = INSTR(B$, "/")
           IF P <> 0 THEN Z = Z + 1: GOTO FindZ
         GOTO Rfirst

Rlop:    LINE INPUT #1, A$

Rfirst:    K = Z: I = 1
           P = INSTR(A$, "/")
           IF P = 0 THEN GOTO RlopE
             A$ = MID$(A$, P + 1)
Rlop1:       V = ASC(A$): A$ = MID$(A$, 2)
                'Allow Sivy's Letters
                IF V = 86 THEN V = 49 'V=1
                IF V = 76 THEN V = 50 'L=2
                IF V = 84 THEN V = 51 'T=3
                IF V = 90 THEN V = 52 'Z=4
                IF V = 65 THEN V = 53 'A=5
                IF V = 66 THEN V = 54 'B=6
                IF V = 80 THEN V = 55 'P=7

                IF V = 68 THEN V = 57 'D=9  SOMA+PLUS
                IF V = 73 THEN V = 57 'I=9
                IF V = 81 THEN V = 57 'Q=9
                IF V = 83 THEN V = 57 'S=9

             IF V >= 49 AND V <= 57 THEN Fld(I, J, K) = V - 48: I = I + 1 '1-9
             IF V = 48 OR V = 45 OR V = 46 OR V = 35 THEN I = I + 1 ' 0.-# are
empty X=X+1
             IF V = 47 THEN X = I: I = 1: K = K - 1                ' /  then X=1
Z=Z-1
             IF A$ <> "" AND K >= 1 THEN GOTO Rlop1
```

```
            J = J + 1
         GOTO Rlop
RlopE:     FigX = X - 1
           FigY = J - 1
           FigZ = Z
         RETURN

'#####################################################################
'#
'# Save BMP image
'#
'#####################################################################
SaveBMP: GOSUB Clrs
ClrKbd6:  IF INKEY$ <> "" THEN GOTO ClrKbd6 'Clear key buffer
         COLOR 3
         PRINT "Type [Q][Enter] to Exit."
         PRINT "Type [A][Enter] to Batch save BMP."
         PRINT "Press [Esc] to Abort during SAVE."
         PRINT "But note that an aborted file will be incomplete."
         PRINT "We DON't check if the name is in use !"
         PRINT "Type [Enter] to use "; : COLOR 11: PRINT Xx$; ".BMP"

         COLOR 11: INPUT "Enter Filename to use (No extension): ", A$

         IF A$ = "A" OR A$ = "a" THEN BatBMP = 1: GOTO BatBMP1
         IF A$ = "Q" OR A$ = "q" THEN GOSUB Clrs: RETURN
         IF INSTR(A$, ".") <> 0 THEN GOTO ClrKbd6
         Ofile$ = Xx$ + ".BMP"
         IF A$ <> "" THEN Ofile$ = A$ + ".BMP"
         GOTO BatBMP2

BatBMP1: IF Aeof = 2 THEN
           Aeof = 0
           FOR N = miny TO maxy     'Erase graphic area
             LINE (minx, N)-(maxx, N), 15
           NEXT N
           BatBMP = 0
           RETURN'End of file flag
         END IF
         GOSUB Clrs
         LOCATE 5, 1
         PRINT "During SAVE, press [Esc] To abort."
         PRINT "But note that the file will be incomplete."
         Ofile$ = Xx$ + ".BMP"


' 'Gives 'OUT OF STACK SPACE ERROR I dont know why, So I don't do it
' 'Test if file is present
'       A = 0: ON ERROR GOTO SaveE: CLOSE #2
'       OPEN Ofile$ FOR INPUT AS #2
'       GOTO SaveF
'SaveE: A = 1: RESUME NEXT
'SaveF: CLOSE #2
'       ON ERROR GOTO 0
'         IF A = 0 THEN
'           INPUT "File exist, Overwrite (Y)es/(N)o ", A$
'           IF A$ <> "Y" AND A$ <> "y" THEN GOTO SaveBMP
'         END IF

BatBMP2: CLOSE #2
         IF Ofile$ = ".BMP" THEN
             GOSUB Clrs: COLOR 12: LOCATE 10, 1
             PRINT "No Filename."
             BatBMP = 0
             RETURN
         END IF
         OPEN Ofile$ FOR OUTPUT AS #2  'Make a new file
         RESTORE BMPList
Headr:   READ N, A$
         IF A$ = "IMAGE" THEN GOTO SaveIMG
         by = VAL("&H" + A$)
         FOR NN = 1 TO N
           PRINT #2, CHR$(by);
```

```
          NEXT NN
          GOTO Headr

'Bottom line left side first 228 byte (start&end with FF)
SaveIMG: FOR Y = miny TO maxy    '-120  137

          IF INKEY$ = CHR$(27) THEN
            GOSUB Clrs
            BatBMP = 0
            RETURN  '---->Abort saving
          END IF

          LOCATE 15, 1: PRINT "Writing: "; maxy - Y
            PRINT #2, CHR$(255);

'(Screen color)BMPColor
'(0)00 Black  (1)04 DBlue    (2)02 DGreen   (3)06 DCyan
'(4)01 DRed   (5)05 DPurple  (6)03 DYellow  (7)07 Gray
'(8)F8 DGray  (9)FC Blue     (10)FA Green   (11)FE Cyan
'(12)F9 Red   (13)FD Purple  (14)FB Yellow  (15)FF White

            FOR X = minx TO maxx  '-105  120
              A = POINT(X, Y)
              AA = (A AND 2) + (A AND 1) * 4 + (A AND 4) / 4 + (A AND 8) * 31
              '=> AA = A AND 2
              '   IF A AND 1 THEN AA = AA + 4
              '   IF A AND 4 THEN AA = AA + 1
              '   IF A AND 8 THEN AA = AA + 248
              PRINT #2, CHR$(AA);
            NEXT X

            PRINT #2, CHR$(255);
          NEXT Y
          CLOSE #2

 IF BatBMP = 1 THEN GOSUB NextFig: GOTO BatBMP1

          GOSUB Clrs
          RETURN

'X=0-225 Y=0-257  (228 * 258 + 1078[header] => 59902 byte file)
BMPList:
    DATA 1,42,1,4D,1,FE,1,E9    'Last byte in file
    DATA 6,00,1,36,1,04,2,00,1,28,3,00,1,E2,3,00  'X size (Actually 228 byte are
stored)
    DATA 1,02,1,01,2,00         'Y size
    DATA 1,01,1,00,1,08,31,00,1,80,2,00,1,80,3,00,2,80,1,00
    DATA 1,80,3,00,1,80,1,00,1,80,1,00,2,80,2,00,3,C0,1,00
    DATA 1,C0,1,DC,1,C0,1,00,1,F0,1,CA,1,A6,1,00,1,FF,1,FB
    DATA 1,F0,1,00,2,80,1,40,2,00,1,FF,1,80,2,00,2,40,1,00
    DATA 1,A6,1,CA,1,F0,2,00,1,80,1,FF,1,00,2,A0,1,A4,2,00
    DATA 1,40,1,80,1,00,1,FF,1,00,1,80,1,00,1,40,1,00,1,80
    DATA 1,00,1,80,1,40,160,00,160,00,160,00,160,00,160,00,102,00
    DATA 1,F0,1,FB,1,FF,1,00,1,A4,2,A0,1,00,3,80,3,00,1,FF
    DATA 2,00,1,FF,3,00,2,FF,1,00,1,FF,3,00,1,FF,1,00,1,FF
    DATA 1,00,2,FF,2,00,3,FF,1,00
    DATA 1,"IMAGE"

'######################################################################
'#
'# Save the solution (After a call of PostSolve by Solve)
'#
'######################################################################
SaveResult:
          GOSUB Clrs
          CLOSE #2
          LOCATE 14, 1
          COLOR 12
          PRINT "When appending to an existing file, the"
          PRINT "appended data will end at the bottom."
          PRINT "This means that if you append to a figure"
          PRINT "file, YOU must manually edit the file, to"
          PRINT "move the figures to the right position."
```

```
         PRINT
         PRINT "I suggest you save new figures in a new file."

         LOCATE 5, 1
ClrKbd2:  IF INKEY$ <> "" THEN GOTO ClrKbd2 'Clear key buffer
         COLOR 3: PRINT "Type [Q][Enter] to Exit."
                  PRINT "Saving to an existing file will append the data"
                  PRINT "Type [Enter] to use ";
         COLOR 11: PRINT file$; ".HTM"
         COLOR 11: INPUT "Enter Filename to use (No extension): ", A$
         IF A$ = "Q" OR A$ = "q" THEN RETURN
         IF INSTR(A$, ".") <> 0 THEN GOTO ClrKbd2
         Ofile$ = file$ + ".HTM"
         IF A$ <> "" THEN Ofile$ = A$ + ".HTM"

'Test if file is present
       A = 0: ON ERROR GOTO SaveC: CLOSE #2
       OPEN Ofile$ FOR INPUT AS #2
       GOTO SaveD
SaveC: A = 1: RESUME NEXT
SaveD: CLOSE #2
         IF A = 0 THEN
           OPEN Ofile$ FOR APPEND AS #2  'Append to existing file
         ELSE
           OPEN Ofile$ FOR OUTPUT AS #2  'Make a new file
           PRINT #2, "Solutions to SOMA figures from "; file$; ".HTM"
           PRINT #2, "Made by 'Bundgård/McFarren's Solution program."
           PRINT #2, ""
         END IF

         PRINT
         COLOR 3: PRINT "Type [Q][Enter] to Exit."
                  PRINT "Type [Enter] to use the name ";
         COLOR 11: PRINT Xx$
         COLOR 11: INPUT "Enter Figure name: ", A$
         IF A$ = "Q" OR A$ = "q" THEN RETURN
         IF A$ <> "" THEN Xx$ = UCASE$(A$)

         PRINT #2, "/SOMA"; Xx$
         FOR J = 1 TO FigY
           FOR K = FigZ TO 1 STEP -1
             PRINT #2, "/";
             FOR I = 1 TO FigX
               IF Fld(I, J, K) = 0 THEN
                 PRINT #2, ".";
               ELSE
                 PRINT #2, USING "#"; Fld(I, J, K);
               END IF
             NEXT I
           NEXT K
           PRINT #2, ""
         NEXT J
         PRINT #2, "---"
         CLOSE
       RETURN

'######################################################################
'#
'# PreSolve before calling Solve Fld()->A()
'#
'######################################################################
'Move data to Solve array A()
'Set X, Y, Z to figure size
PreSolve:
  F = 0
  FOR Zz = 1 TO 16
    FOR Yy = 1 TO 16
      FOR Xx = 1 TO 16
        IF Fld(Xx, Yy, Zz) <> 0 THEN
           F = F + 1
           A(Xx, Yy, Zz) = F
        ELSE
           A(Xx, Yy, Zz) = 0
```

```
        END IF
      NEXT Xx
    NEXT Yy
  NEXT Zz
  X = FigX
  Y = FigY
  Z = FigZ
  RETURN

'#####################################################################
'#
'# PostSolve. After Solve A()->Fld()
'#
'#####################################################################
'Sort the result into the figure
PostSolve:
GOSUB PreSolve
    FOR K = 1 TO Z
      FOR J = 1 TO Y
        FOR I = 1 TO X
        IF A(I, J, K) <> 0 THEN

        Ex = 0
          FOR A = 1 TO 7
            FOR B = 1 TO 4
              IF Ex = 0 AND AA(A, B) = A(I, J, K) THEN A(I, J, K) = A: Ex = 1
            NEXT B
          NEXT A
        END IF
        NEXT I
      NEXT J
    NEXT K

'Move answer to display array
  FOR Zz = 1 TO 16
    FOR Yy = 1 TO 16
      FOR Xx = 1 TO 16
        Fld(Xx, Yy, Zz) = A(Xx, Yy, Zz)
      NEXT Xx
    NEXT Yy
  NEXT Zz
  RETURN



'#####################################################################
'#
'# Trying to add a symmetric element
'# Thorleif 19.02.1999
'#
'#####################################################################

'Find hiresolution time (UNUSED IN THIS PROGRAM)
Ti:      DEF SEG = 0
         A = PEEK(1135)
         B = PEEK(1134)  '46E
         C = PEEK(1133)
         D = PEEK(1132)  '46C
         OUT (67), 0     'Timer command register
         F = INP(64)     'Counter 0 lo
         E = INP(64)     'Counter 0 hi
         PRINT A; B; C; D; E; F
         'time=A:B:C:D:E:F / 1.19318MHz
         INPUT TT
         GOTO Ti



Symmetry:


'XXX Xsymmetry XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'(FigX is always>=2)
```

```
  FOR N = 1 TO 27
    SymUse(N) = 0             'Set cubes for No Use
  NEXT N

  Xsy% = INT((FigX + 1) / 2)'Half size, +1 if ODD
  Xsm% = FigX + 1            'Max size+1
  FOR Zz% = 1 TO FigZ
    FOR Yy% = 1 TO FigY
      FOR Xx% = 1 TO Xsy%
        N% = A(Xx%, Yy%, Zz%)
          SymUse(N%) = 1      'Checked cell, Unassigned sets SymUse(0)=1
                             'SGN + SGN = 0 or 2 if Symmetry
          IF SGN(N%) + SGN(A(Xsm% - Xx%, Yy%, Zz%)) = 1 THEN GOTO NoX 'No X
symmetry
      NEXT Xx%
    NEXT Yy%
  NEXT Zz%

'Now remove elements if symmetry
'We can only remove elements of one piece, because we
'dont know if the other elements land in the same symmetric side.
'We remove piece 1 elements

    J = 1: D = 1
RemoX:
    A = 0
    FOR K = 1 TO 3
      A = A + SymUse(S(1, J, K)) 'How many cells are in use
    NEXT K
    IF A >= 2 THEN 'At least 2 cells in use
      S(1, D, 1) = S(1, J, 1)
      S(1, D, 2) = S(1, J, 2)
      S(1, D, 3) = S(1, J, 3)
      D = D + 1
    END IF
    J = J + 1
    IF S(1, J, 1) <> 0 AND J <= 160 THEN GOTO RemoX
      S(1, D, 1) = 0
      S(1, D, 2) = 0
      S(1, D, 3) = 0

    Xsym = 1
    LOCATE 13, 1
      PRINT "X Symmetry removed"; J - D; "elements of"; J - 1; "tests."
    GOTO XsyEnd

NoX:
    Xsym = 0
    LOCATE 13, 1
    PRINT "No X symmetry."

XsyEnd:
    'RETURN



'YYY Ysymmetry YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
'(FigY is always>=2)

  FOR N = 1 TO 27
    SymUse(N) = 0             'Set cubes for No Use
  NEXT N

  Ysy% = INT((FigY + 1) / 2)'Half size, +1 if ODD
  Ysm% = FigY + 1            'Max size+1
  FOR Zz% = 1 TO FigZ
    FOR Xx% = 1 TO FigX
      FOR Yy% = 1 TO Ysy%
        N% = A(Xx%, Yy%, Zz%)
          SymUse(N%) = 1      'Checked cell, Unassigned sets SymUse(0)=1
                             'SGN + SGN = 0 or 2 if Symmetry
          IF SGN(N%) + SGN(A(Xx%, Ysm% - Yy%, Zz%)) = 1 THEN GOTO NoY'No Y
```

```
symmetry
      NEXT Yy%
    NEXT Xx%
  NEXT Zz%

    J = 1: D = 1
RemoY:
    A = 0
    FOR K = 1 TO 3
      A = A + SymUse(S(1, J, K)) 'How many cells are in use
    NEXT K
    IF A >= 2 THEN 'At least 2 cells in use
      S(1, D, 1) = S(1, J, 1)
      S(1, D, 2) = S(1, J, 2)
      S(1, D, 3) = S(1, J, 3)
      D = D + 1
    END IF
    J = J + 1
    IF S(1, J, 1) <> 0 AND J <= 160 THEN GOTO RemoY
      S(1, D, 1) = 0
      S(1, D, 2) = 0
      S(1, D, 3) = 0

    Ysym = 1
    LOCATE 14, 1
      PRINT "Y Symmetry removed"; J - D; "elements of"; J - 1; "tests."
    GOTO YsyEnd

NoY:
    Ysym = 0
    LOCATE 14, 1
    PRINT "No Y symmetry."

YsyEnd:
    RETURN



'SSS End SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS




'#####################################################################
'#
'# Find a solution to the SOMA figure
'# Courtney McFarren
'# 1999.02.09:19.30 TB Patched the revised ANSWER5 code, added a few comments
'# 1999.02.15       CMF Converted all CUB file references to S(,,)
'#
'#####################################################################

FindSolve:
     LOCATE 1, 1
     PRINT "Now computing positions"
     Tim! = TIMER
         GOSUB solve
         IF EFLAG = 1 THEN solve = 0: RETURN
         GOSUB SArray
     Tims! = TIMER
         GOSUB Symmetry
     Timsx! = TIMER
         GOSUB IslPar
         GOSUB Answer
     Timen! = TIMER
     LOCATE 20, 1
     PRINT USING "Solve time: ##.## Sec."; Timen! - Tim!
     PRINT USING "Sym proces: ##.## Sec."; Timsx! - Tims!
     RETURN

'### SOLVE.BAS ############################################

solve:
```

```
  Xxmin = 16: Xxmax = 0: Yymin = 16: Yymax = 0: Zzmin = 16: Zzmax = 0
  FOR Zz = 1 TO 16
    FOR Yy = 1 TO 16
      FOR Xx = 1 TO 16
        IF A(Xx, Yy, Zz) <> 0 THEN
        IF Xx < Xxmin THEN Xxmin = Xx
        IF Xx > Xxmax THEN Xxmax = Xx
        IF Yy < Yymin THEN Yymin = Yy
        IF Yy > Yymax THEN Yymax = Yy
        IF Zz < Zzmin THEN Zzmin = Zz
        IF Zz > Zzmax THEN Zzmax = Zz
        END IF
      NEXT Xx
    NEXT Yy
  NEXT Zz
  FigX = Xxmax - Xxmin + 1: X = FigX
  FigY = Yymax - Yymin + 1: Y = FigY
  FigZ = Zzmax - Zzmin + 1: Z = FigZ

  F7 = 0: G = 0: EFLAG = 1
  FOR K = 1 TO Z
    FOR J = 1 TO Y
      FOR I = 1 TO X
      IF A(I, J, K) THEN
        F7 = F7 + 1
        G = G + (I + J + K) MOD 2
      END IF
      NEXT I
    NEXT J
  NEXT K

  F4 = F7 \ 4: F3 = F7 MOD 4
  IF F3 = 1 OR F3 = 2 THEN PRINT "ERROR ON CUBE COUNT": RETURN
  IF F7 = 0 OR F7 > 27 THEN PRINT "ERROR ON CUBE COUNT": RETURN
  G = ABS(G * 2 - F7)
  IF G > 5 THEN PRINT "PARITY ERROR; NO SOL'N EXISTS": RETURN
  EFLAG = 0
  RETURN '---->GOTO SArray


'### SARRAY.BAS ##############################################

SArray:

  SWAP X0, X: SWAP Y0, Y: SWAP Z0, Z
  FOR I = 1 TO 7: FOR J = 1 TO 160: FOR K = 1 TO 4
  S(I, J, K) = 0: NEXT K, J, I
  RESTORE SarDat

  Q = 4: D0 = 1: D1 = 0
  FOR I = 1 TO Q: READ F(I): NEXT I
  GOSUB Sar001 'piece #1

  Q = 4: D0 = 2: D1 = 0
  FOR I = 1 TO Q: READ F(I): NEXT I
  GOSUB Sar234 'piece #2
  Q = 2: D0 = 3: D1 = 0
  FOR I = 1 TO Q: READ F(I): NEXT I
  GOSUB Sar234 'piece#3
  Q = 2: D0 = 4: D1 = 0
  FOR I = 1 TO Q: READ F(I): NEXT I
  GOSUB Sar234 'piece #4

  Q = 12: D0 = 5: D1 = 0
  FOR I = 1 TO Q: READ F(I): NEXT I
  GOSUB Sar567 'piece #5
  Q = 12: D0 = 6: D1 = 0
  FOR I = 1 TO Q: READ F(I): NEXT I
  GOSUB Sar567 'piece #6
  Q = 8: D0 = 7: D1 = 0
  FOR I = 1 TO Q: READ F(I): NEXT I
  GOSUB Sar567 'piece #7
```

```
  SWAP X, X0: SWAP Y, Y0: SWAP Z, Z0
  RETURN '----> GOTO IslPar

Sar001: 'calculate positions for piece #1 (2x2x1)

  X1 = X0 - 1: Y1 = Y0 - 1: Z1 = Z0
  X2 = X0 - 1: Y2 = Y0: Z2 = Z0 - 1
  X3 = X0: Y3 = Y0 - 1: Z3 = Z0 - 1

  'xy plane

    FOR K = 1 TO Z1: FOR J = 1 TO Y1: FOR I = 1 TO X1
      L = -1: M = 0
      FOR Y = 0 TO 1: FOR X = 0 TO 1
        N = SGN(A(I + X, J + Y, K))
        L = L + 1: M = M + N * 2 ^ L
      NEXT X, Y
      FOR W = 1 TO Q
        IF ((F(W) AND M) = F(W)) THEN
          Q1 = 0: Q2 = F(W)
          FOR Y = 0 TO 1: FOR X = 0 TO 1
            Q3 = Q2 MOD 2
            IF Q3 = 1 THEN Q1 = Q1 + 1: C(Q1) = A(I + X, J + Y, K)
            Q2 = Q2 \ 2
          NEXT X, Y: D1 = D1 + 1
          FOR W1 = 1 TO 3: S(D0, D1, W1) = C(W1): NEXT W1
        END IF
      NEXT W
    NEXT I, J, K

  'xy plane

    FOR K = 1 TO Z2: FOR J = 1 TO Y2: FOR I = 1 TO X2
      L = -1: M = 0
      FOR Y = 0 TO 1: FOR X = 0 TO 1
        N = SGN(A(I + X, J, K + Y))
        L = L + 1: M = M + N * 2 ^ L
      NEXT X, Y
      FOR W = 1 TO Q
        IF (F(W) AND M) = F(W) THEN
          Q1 = 0: Q2 = F(W)
          FOR Y = 0 TO 1: FOR X = 0 TO 1
            Q3 = Q2 MOD 2
            IF Q3 = 1 THEN Q1 = Q1 + 1: C(Q1) = A(I + X, J, K + Y)
            Q2 = Q2 \ 2
          NEXT X, Y: D1 = D1 + 1
          FOR W1 = 1 TO 3: S(D0, D1, W1) = C(W1): NEXT W1
        END IF
      NEXT W
    NEXT I, J, K

  'yz plane

    FOR K = 1 TO Z3: FOR J = 1 TO Y3: FOR I = 1 TO X3
      L = -1: M = 0
      FOR Y = 0 TO 1: FOR X = 0 TO 1
        N = SGN(A(I, J + X, K + Y))
        L = L + 1: M = M + N * 2 ^ L
      NEXT X, Y
      FOR W = 1 TO Q
        IF (F(W) AND M) = F(W) THEN
          Q1 = 0: Q2 = F(W)
          FOR Y = 0 TO 1: FOR X = 0 TO 1
            Q3 = Q2 MOD 2
            IF Q3 = 1 THEN Q1 = Q1 + 1: C(Q1) = A(I, J + X, K + Y)
            Q2 = Q2 \ 2
          NEXT X, Y: D1 = D1 + 1
          FOR W1 = 1 TO 3: S(D0, D1, W1) = C(W1): NEXT W1
        END IF
      NEXT W
    NEXT I, J, K
```

```
    RETURN '---->GOTO Symmetry/IslPar

Sar234: 'calculate positions for pieces #2, 3 &4 (3x2x1)

  X1 = X0 - 2: Y1 = Y0 - 1: Z1 = Z0
  X2 = X0 - 1: Y2 = Y0 - 2: Z2 = Z0
  X3 = X0 - 2: Y3 = Y0: Z3 = Z0 - 1
  X4 = X0: Y4 = Y0 - 2: Z4 = Z0 - 1
  X5 = X0 - 1: Y5 = Y0: Z5 = Z0 - 2
  X6 = X0: Y6 = Y0 - 1: Z6 = Z0 - 2

  '~~~~xy plane~~~~

    FOR K = 1 TO Z1: FOR J = 1 TO Y1: FOR I = 1 TO X1
      L = -1: M = 0
      FOR Y = 0 TO 1: FOR X = 0 TO 2
        N = SGN(A(I + X, J + Y, K))
        L = L + 1: M = M + N * 2 ^ L
      NEXT X, Y
      FOR W = 1 TO Q
        IF ((F(W) AND M) = F(W)) THEN
          Q1 = 0: Q2 = F(W)
          FOR Y = 0 TO 1: FOR X = 0 TO 2
            Q3 = Q2 MOD 2
            IF Q3 = 1 THEN Q1 = Q1 + 1: C(Q1) = A(I + X, J + Y, K)
            Q2 = Q2 \ 2
          NEXT X, Y: D1 = D1 + 1
          FOR W1 = 1 TO 4: S(D0, D1, W1) = C(W1): NEXT W1
        END IF
      NEXT W
    NEXT I, J, K

    FOR K = 1 TO Z2: FOR J = 1 TO Y2: FOR I = 1 TO X2
      L = -1: M = 0
      FOR Y = 0 TO 1: FOR X = 0 TO 2
        N = SGN(A(I + Y, J + X, K))
        L = L + 1: M = M + N * 2 ^ L
      NEXT X, Y
      FOR W = 1 TO Q
        IF ((F(W) AND M) = F(W)) THEN
          Q1 = 0: Q2 = F(W)
          FOR Y = 0 TO 1: FOR X = 0 TO 2
            Q3 = Q2 MOD 2
            IF Q3 = 1 THEN Q1 = Q1 + 1: C(Q1) = A(I + Y, J + X, K)
            Q2 = Q2 \ 2
          NEXT X, Y: D1 = D1 + 1
          FOR W1 = 1 TO 4: S(D0, D1, W1) = C(W1): NEXT W1
        END IF
      NEXT W
    NEXT I, J, K

  '~~~~xz plane~~~~

    FOR K = 1 TO Z3: FOR J = 1 TO Y3: FOR I = 1 TO X3
      L = -1: M = 0
      FOR Y = 0 TO 1: FOR X = 0 TO 2
        N = SGN(A(I + X, J, K + Y))
        L = L + 1: M = M + N * 2 ^ L
      NEXT X, Y
      FOR W = 1 TO Q
        IF ((F(W) AND M) = F(W)) THEN
          Q1 = 0: Q2 = F(W)
          FOR Y = 0 TO 1: FOR X = 0 TO 2
            Q3 = Q2 MOD 2
            IF Q3 = 1 THEN Q1 = Q1 + 1: C(Q1) = A(I + X, J, K + Y)
            Q2 = Q2 \ 2
          NEXT X, Y: D1 = D1 + 1
          FOR W1 = 1 TO 4: S(D0, D1, W1) = C(W1): NEXT W1
        END IF
      NEXT W
    NEXT I, J, K

    FOR K = 1 TO Z4: FOR J = 1 TO Y4: FOR I = 1 TO X4
```

```
      L = -1: M = 0
      FOR Y = 0 TO 1: FOR X = 0 TO 2
        N = SGN(A(I, J + X, K + Y))
        L = L + 1: M = M + N * 2 ^ L
      NEXT X, Y
      FOR W = 1 TO Q
        IF ((F(W) AND M) = F(W)) THEN
          Q1 = 0: Q2 = F(W)
          FOR Y = 0 TO 1: FOR X = 0 TO 2
            Q3 = Q2 MOD 2
            IF Q3 = 1 THEN Q1 = Q1 + 1: C(Q1) = A(I, J + X, K + Y)
            Q2 = Q2 \ 2
          NEXT X, Y: D1 = D1 + 1
          FOR W1 = 1 TO 4: S(D0, D1, W1) = C(W1): NEXT W1
        END IF
      NEXT W
    NEXT I, J, K

  '~~~~yz plane~~~~

    FOR K = 1 TO Z5: FOR J = 1 TO Y5: FOR I = 1 TO X5
      L = -1: M = 0
      FOR Y = 0 TO 1: FOR X = 0 TO 2
        N = SGN(A(I + Y, J, K + X))
        L = L + 1: M = M + N * 2 ^ L
      NEXT X, Y
      FOR W = 1 TO Q
        IF ((F(W) AND M) = F(W)) THEN
          Q1 = 0: Q2 = F(W)
          FOR Y = 0 TO 1: FOR X = 0 TO 2
            Q3 = Q2 MOD 2
            IF Q3 = 1 THEN Q1 = Q1 + 1: C(Q1) = A(I + Y, J, K + X)
            Q2 = Q2 \ 2
          NEXT X, Y: D1 = D1 + 1
          FOR W1 = 1 TO 4: S(D0, D1, W1) = C(W1): NEXT W1
        END IF
      NEXT W
    NEXT I, J, K

    FOR K = 1 TO Z6: FOR J = 1 TO Y6: FOR I = 1 TO X6
      L = -1: M = 0
      FOR Y = 0 TO 1: FOR X = 0 TO 2
        N = SGN(A(I, J + Y, K + X))
        L = L + 1: M = M + N * 2 ^ L
      NEXT X, Y
      FOR W = 1 TO Q
        IF ((F(W) AND M) = F(W)) THEN
          Q1 = 0: Q2 = F(W)
          FOR Y = 0 TO 1: FOR X = 0 TO 2
            Q3 = Q2 MOD 2
            IF Q3 = 1 THEN Q1 = Q1 + 1: C(Q1) = A(I, J + Y, K + X)
            Q2 = Q2 \ 2
          NEXT X, Y: D1 = D1 + 1
          FOR W1 = 1 TO 4: S(D0, D1, W1) = C(W1): NEXT W1
        END IF
      NEXT W
    NEXT I, J, K

    RETURN

Sar567: 'calculate positions for pieces #5, 6 &7 (2x2x2)

    X1 = X0 - 1: Y1 = Y0 - 1: Z1 = Z0 - 1
    FOR K = 1 TO Z1: FOR J = 1 TO Y1: FOR I = 1 TO X1
      L = -1: M = 0
      FOR Z = 0 TO 1: FOR Y = 0 TO 1: FOR X = 0 TO 1
        N = SGN(A(I + X, J + Y, K + Z))
        L = L + 1: M = M + N * 2 ^ L
      NEXT X, Y, Z
      FOR W = 1 TO Q
        IF ((F(W) AND M) = F(W)) THEN
          Q1 = 0: Q2 = F(W)
          FOR Z = 0 TO 1: FOR Y = 0 TO 1: FOR X = 0 TO 1
```

```
            Q3 = Q2 MOD 2
            IF Q3 = 1 THEN Q1 = Q1 + 1: C(Q1) = A(I + X, J + Y, K + Z)
            Q2 = Q2 \ 2
          NEXT X, Y, Z: D1 = D1 + 1
          FOR W1 = 1 TO 4: S(D0, D1, W1) = C(W1): NEXT W1
        END IF
      NEXT W
    NEXT I, J, K
    RETURN

SarDat: 'binary data for pieces 1 through 7

  DATA 7,11,13,14
  DATA 15,39,57,60
  DATA 23,58
  DATA 30,51
  DATA 27,46,53,71,92,114,141,163,184,202,209,228
  DATA 29,39,58,78,83,116,139,172,177,197,216,226
  DATA 23,43,77,113,142,178,212,232


'### ISLPAR.BAS ##########################################

IslPar:

'~~~~define the island matrix~~~~

  FOR I = 1 TO 27: FOR J = 0 TO 6: E0(I, J) = 0: NEXT J, I
  FOR K = 1 TO Z: FOR J = 1 TO Y: FOR I = 1 TO X
    IF A(I, J, K) THEN
      L = A(I, J, K): E0(L, 0) = L
      E0(L, 1) = A(I - 1, J, K)
      E0(L, 2) = A(I + 1, J, K)
      E0(L, 3) = A(I, J - 1, K)
      E0(L, 4) = A(I, J + 1, K)
      E0(L, 5) = A(I, J, K - 1)
      E0(L, 6) = A(I, J, K + 1)
    END IF
  NEXT I, J, K

'~~~~define the parity string~~~~

  FOR I = 1 TO 27: G(I) = 0: NEXT I
  FOR K = 1 TO Z: FOR J = 1 TO Y: FOR I = 1 TO X
    G0 = A(I, J, K)
    IF G0 THEN G(G0) = (I + J + K) MOD 2
  NEXT I, J, K
  FOR I = 1 TO 27: G(I) = G(I) * 2 - 1: NEXT I
  G0 = 0: FOR I = 1 TO 27: G0 = G0 + G(I): NEXT I
  PARITY = G0: G0 = ((G0 + 5) \ 2) MOD 2
  IF G0 THEN PAR = 1 ELSE PAR = -1

'~~~~fix the S-array~~~~

  IF F7 = 27 THEN RETURN
  G = 27: F6 = 6 - F4
  IF F3 = 0 THEN
    S(1, 1, 1) = 27: S(1, 1, 2) = 26: S(1, 1, 3) = 25: G = 24
    FOR I = 2 TO 160: FOR J = 1 TO 4: S(1, I, J) = 0: NEXT J, I
  END IF
  IF F4 = 0 THEN
    FOR I = 2 TO 7: FOR J = 1 TO 160: FOR K = 1 TO 4
    S(I, J, K) = 0: NEXT K, J, I
    FOR J = 1 TO 6
      FOR I = 2 TO 7: FOR K = 1 TO 4
        S(I, J, K) = G - K + 1
      NEXT K, I
      G = G - 4
    NEXT J
  ELSEIF F4 = 6 THEN
    REM
  ELSE
    FOR I = 2 TO 7: FOR J = (160 - F6) TO 1 STEP -1: FOR K = 1 TO 4
```

```
      S(I, J + F6, K) = S(I, J, K)
    NEXT K, J, I
    FOR J = 1 TO F6
      FOR I = 2 TO 7: FOR K = 1 TO 4
        S(I, J, K) = G - K + 1
      NEXT K, I: G = G - 4
    NEXT J
  END IF

  RETURN'----> GOTO Answer


'### ANSWER.BAS ##########################################

Answer:

      FOR I = 1 TO 27: B(I) = 1: NEXT I
      FOR I = 1 TO 4: C(I) = 0: NEXT I
      FOR I = 1 TO 7: D(I) = 0: NEXT I
      FOR I = 1 TO 7: FOR J = 1 TO 4: AA(I, J) = 0: NEXT J, I

PC1: '11111111111111111111111111

      D(1) = D(1) + 1: IF S(1, D(1), 1) = 0 THEN GOTO NoSoln
      FOR I = 1 TO 3: C(I) = S(1, D(1), I): NEXT I: C(4) = 0
      GOSUB ParChk: IF ABORT THEN GOTO PC1 'parity check
      GOSUB IslChk: IF ABORT THEN GOTO PC1 'island check
      FOR I = 1 TO 3: AA(1, I) = C(I): B(C(I)) = 0: NEXT I
      LOCATE 3, 1: PRINT 1; D(1)

PC7: '77777777777777777777777777

      D(7) = D(7) + 1
      IF S(7, D(7), 1) = 0 THEN
        FOR I = 1 TO 3: B(AA(1, I)) = 1: NEXT I
        D(7) = 0: GOTO PC1
      END IF: I = 0: DO
      I = (I + 1) MOD 5
      C(I) = S(7, D(7), I)
      LOOP UNTIL B(C(I)) = 0
      IF I THEN GOTO PC7
      GOSUB IslChk: IF ABORT THEN GOTO PC7
      FOR I = 1 TO 4: AA(7, I) = C(I): B(C(I)) = 0: NEXT I
      LOCATE 9, 1: PRINT 7; D(7)

PC3: '33333333333333333333333333

      D(3) = D(3) + 1
      IF S(3, D(3), 1) = 0 THEN
        FOR I = 1 TO 4: B(AA(7, I)) = 1: NEXT I
        D(3) = 0: GOTO PC7
      END IF: I = 0: DO
      I = (I + 1) MOD 5
      C(I) = S(3, D(3), I)
      LOOP UNTIL B(C(I)) = 0
      IF I THEN GOTO PC3
      GOSUB IslChk: IF ABORT THEN GOTO PC3
      FOR I = 1 TO 4: AA(3, I) = C(I): B(C(I)) = 0: NEXT I
      LOCATE 5, 1: PRINT 3; D(3)

PC4: '44444444444444444444444444

      D(4) = D(4) + 1
      IF S(4, D(4), 1) = 0 THEN
        FOR I = 1 TO 4: B(AA(3, I)) = 1: NEXT I
        D(4) = 0: GOTO PC3
      END IF: I = 0: DO
      I = (I + 1) MOD 5
      C(I) = S(4, D(4), I)
      LOOP UNTIL B(C(I)) = 0
      IF I THEN GOTO PC4
      GOSUB IslChk: IF ABORT THEN GOTO PC4
      FOR I = 1 TO 4: AA(4, I) = C(I): B(C(I)) = 0: NEXT I
```

```
      LOCATE 6, 1: PRINT 4; D(4)

PC5: '55555555555555555555555555

      D(5) = D(5) + 1
      IF S(5, D(5), 1) = 0 THEN
        FOR I = 1 TO 4: B(AA(4, I)) = 1: NEXT I
        D(5) = 0: GOTO PC4
      END IF: I = 0: DO
      I = (I + 1) MOD 5
      C(I) = S(5, D(5), I)
      LOOP UNTIL B(C(I)) = 0
      IF I THEN GOTO PC5
      GOSUB IslChk: IF ABORT THEN GOTO PC5
      FOR I = 1 TO 4: AA(5, I) = C(I): B(C(I)) = 0: NEXT I
      LOCATE 7, 1: PRINT 5; D(5)

PC6: '66666666666666666666666666

      D(6) = D(6) + 1
      IF S(6, D(6), 1) = 0 THEN
        FOR I = 1 TO 4: B(AA(5, I)) = 1: NEXT I
        D(6) = 0: GOTO PC5
      END IF: I = 0: DO
      I = (I + 1) MOD 5
      C(I) = S(6, D(6), I)
      LOOP UNTIL B(C(I)) = 0
      IF I THEN GOTO PC6
      GOSUB IslChk: IF ABORT THEN GOTO PC6
      FOR I = 1 TO 4: AA(6, I) = C(I): B(C(I)) = 0: NEXT I
      LOCATE 8, 1: PRINT 6; D(6)

PC2: '22222222222222222222222222

      D(2) = D(2) + 1
      IF S(2, D(2), 1) = 0 THEN
        FOR I = 1 TO 4: B(AA(6, I)) = 1: NEXT I
        D(2) = 0: GOTO PC6
      END IF: I = 0: DO
      I = (I + 1) MOD 5
      C(I) = S(2, D(2), I)
      LOOP UNTIL B(C(I)) = 0
      IF I THEN GOTO PC2
      FOR I = 1 TO 4: AA(2, I) = C(I): B(C(I)) = 0: NEXT I
      LOCATE 4, 1: PRINT 2; D(2)

'~~~~~~~~~~~~~~~~~~~~~~~~~~

SolnEx:

      COLOR 11
      LOCATE 3, 1: PRINT "SOLUTION NR: "; Reca; " EXISTS!"
      PRINT "        "
      FOR I = 1 TO 7: LOCATE I + 4, 1: PRINT STR$(I); : PRINT ": ";
        IF AA(I, 1) <= F7 THEN
          FOR J = 1 TO 4
            IF AA(I, J) THEN PRINT USING "####"; AA(I, J);
          NEXT J: PRINT
          ELSE PRINT "    "
        END IF
      NEXT I
      solve = 1 'Flag that we succeeded
      RETURN

'Find next possible solution
Nxtsolv:
      FOR I = 1 TO 4: B(AA(2, I)) = 1: NEXT I
      GOTO PC2

NoSoln:
      LOCATE 11, 1
      COLOR 12
      PRINT "SORRY, NO SOLUTION EXISTS"
```

```
      solve = 0
      NF = 1
      RETURN

IslChk:  'island check ++++++++++++++++++++++++++

          ABORT = 0
          IF C(1) > F7 THEN RETURN
          FOR I = 1 TO 27
            IF B(I) THEN E1(I) = E0(I, 0) ELSE E1(I) = 0
          NEXT I
          FOR I = 1 TO 4: E1(C(I)) = 0: NEXT I
          FOR I = 1 TO 60: E(I) = 0: NEXT I

          P1 = 1: P2 = 1
          FOR P0 = 1 TO 27
            N = E1(P0)
            IF N THEN
              E(P1) = N: E1(N) = 0
              DO WHILE N <> 0
                FOR I = 1 TO 6
                  M = E0(N, I)
                  IF M THEN
                    IF E1(M) THEN
                      P2 = P2 + 1: E(P2) = M: E1(M) = 0
                    END IF
                  END IF
                NEXT I
                P1 = P1 + 1: N = E(P1)
              LOOP
              P1 = P1 + 1: P2 = P1
            END IF
          NEXT P0

          P1 = 1: M = 0: N = 0
          DO WHILE N = 0
            DO WHILE E(P1) > 0
              M = M + 1: P1 = P1 + 1
            LOOP
            N = M MOD 4: P1 = P1 + 1
            IF E(P1) = 0 THEN RETURN
          LOOP
          ABORT = 1: RETURN

ParChk:  'parity check ++++++++++++++++++++++++++

          ABORT = 0
          IF F7 < 27 THEN RETURN
          G0 = G(C(1)) + G(C(2)) + G(C(3)) - PAR
          IF G0 <> 0 THEN ABORT = 1
          RETURN


'#####################################################################
'     ---=== END ===---
```