HP-71B Sudoku Generator ... & Coach !

Valentín Albillo (#1075, PPC #4747)

After my original article ***HP-71B Short & Sweet Sudoku Solver*** first appeared in Datafile V24N2P22, shortly followed by the modestly titled ***HP-71B Sudoku Solver's Sublime Sequel*** featured in V24N3P23, I moved on to worthiest pastures, considering my brief but intense Sudoku involvement finished for good. But lo and behold, my wife took a liking to Sudoku as well, and duly began to solve newspaper puzzles and such. Some would resist her best efforts and I then proceeded to proudly show her my solver program. To my utter amazement, she was less than impressed because what she wanted was for my program to actually *generate* puzzles for her of a specific, selectable difficulty, and further she didn't want the whole solution being thrown at her like that, which would only ruin her fun. She only wanted a few *hints* when the moment came where she would find herself hopelessly stuck. In other words, she actually had little use for my full solver, what she actually longed for was a Sudoku ***generator*** and ***coach***, which would supply puzzles and provide hints on demand.

Easier said than done, of course, but having been given a worthwhile goal in life, I immediately set up to the task of creating a Sudoku Generator & Coach for my faithful **HP-71B** [1] and so this very article and program came to be.

What it does do

SUDOGEN is a petite program (only 88 lines = 3,267 bytes without comments) yet it was designed to meet all of the following requirements:

- It can generate *a virtually endless* supply of puzzles of user-selectable difficulty by allowing the user to specify the number of blanks to fill up as well as its solvability type. Puzzles can be generated completely *at random* or in a *repeatable* way, by issuing a previous **`RANDOMIZE n`** statement.
- It can generate puzzles guaranteed to have a *unique solution* which can be *logically deducted* without ever having to guess, or optionally only guaranteed to have at least one legal solution (which is faster).
- The generated puzzles can optionally be *symmetric* if desired.
- It will *print* or *display* the generated puzzle and, optionally, the fully solved puzzle at the time of generation, either pretty-printed or in compact form.
- It can print the puzzle (and solution) in *80-column mode* in an HP-IL printer or display (physical or emulated), so that the user can have a paper copy to use while solving it with pencil and eraser. It can also output to the built-in 22-character display, in which case a compact output format is used.

[1] No HP-71B, HP-IL ROM or Math ROM ? No problem. Google for **Emu71**, a free emulator for Windows (>200X faster) or **HP-71X**, an excellent emulator (>3X) for your HP48/49.

- It can offer *coaching* for the puzzles it generates, giving *commented hints one at a time*, till the whole puzzle is solved, no more hints can be given, or no more hints are requested.
- It will also try to solve and offer *coaching* for *externally generated puzzles* that the user inputs, taken from a newspaper, say.
- It is designed to be as fast as possible, subject to meeting the above requirements. Nevertheless, running it under an emulator such as **Emu71** or **HP-71X** is highly recommended to maximize the enjoyment and usefulness.

How it does do it

The featured program listing does include relevant comments, but here we'll have a general look at the way the program accomplishes its many goals:

The built-in Solver:

In order to successfully generate sufficiently random puzzles, **SUDOGEN** includes a built-in solver to test the solvability of the generated puzzles and, optionally, certify that the solution is *unique* and can be found *by logical deduction alone*, without the user ever being forced to guess.

As it must repeatedly call the solver while generating a puzzle and this would result in extremely long generation times, my full recursive solver isn't used but a simplified non-recursive version which has been fine-tuned for speed and further enhanced to allow for *coaching*, being capable of solving the puzzle on a cell-by-cell basis, as well as finding out and returning a list of all values that are legal for each and every unoccupied cell, thus featuring the capability of giving hints on request.

To that effect, the built-in solver refrains from using recursion and tries instead to fill up empty cells by finding out which digits are unique for a given cell and which cells are unique for a given digit, as fully explained in my referenced articles. This procedure, suitably iterated, will produce a list of unique digits for every location whose contents are thus forced, and a list of all possible legal digits for cells that still admit more than one at the end of the deductive procedure.

These solver capabilities are used in two different ways:

- If the user has specified checking the solvability of the generated puzzle (i.e., it has a unique solution which can be logically deducted without guessing, as stated above) then, after generating a tentative puzzle, **SUDOGEN** calls the solver to test that the puzzle is indeed solvable under these conditions. If the puzzle won't pass muster, it is discarded and another puzzle is promptly generated and tested. Else, the solvable puzzle is output.

- If the user has requested coaching, the solver is called to generate a single *hint*, which will be offered to the user. The user can then go on with their own puzzle-solving and/or request another hint. See below for details of the hinting mechanism and types of hints.

The Puzzle Generator:

As stated, the puzzle generator must try and get a *solvable* puzzle which meets the user's requirements, and must do it in non-geological times. The usual approach in other generators is to proceed either recursively or by using some backtracking mechanism, where the cells are assigned values one by one, with some randomness being present as well. This would be extremely slow here, specially if trying to guarantee *uniqueness* of the solution and that solving it must not require guessing on the part of the user but can be logically deducted.

This being so, **SUDOGEN** takes a wholly different approach. It can be mathematically proved that there are 6,670,903,752,021,072,936,960 valid Sudoku grids, but every valid grid belongs to some so-called *equivalence class*. Two grids are in the same equivalence class if one can be transformed into the other using the invariant transformations that make one puzzle mathematically equivalent to another, and grids in the same equivalence class can be considered as the same grid to all effects. This means that the number of essentially *different* Sudoku grids is many orders of magnitude smaller, only 5,472,730,538 in fact. And conversely, each and every valid grid can be generated from its equivalence class by applying some invariant transformations to it.

SUDOGEN's approach takes advantage of the above to generate an essentially indefinite number of grids starting from any number of pre-given, distinct equivalence classes, which are available to the program in encoded form as `DATA` statements. For simplicity, just three are included in the featured listing, but more can be used by simply inserting additional `DATA` statements and updating the appropriate constants. Whatever their number, the generator will chose at random among them and the chosen equivalence class will then be subject to random transformations till a grid which meets the user's requirements is produced. Due to the randomization, the chances of generating the same puzzle twice is *nil*, and due to the intrinsic characteristics of the transformations being applied, the chance of the user recognizing the particular equivalence class or details of its solution is *nil* as well, being far, far easier to try and solve the puzzle than to guess any details at all of its solution from past experiences. The generation proceeds as follows:

- An equivalence class is selected at random and *decoded*.
- A random *relabelling* is generated and all entries in the equivalence class are relabelled according to it.

- Grid rows are randomly and *invariantly* swapped, then columns.
- Randomly, the whole grid is *transposed*: rows become columns, etc.
- The resulting random valid grid is then *decimated* to empty the number of blanks requested by the user, thus allowing for different difficulty levels by varying the number of blanks to fill up. The decimation can proceed either fully at random or following a random symmetric pattern if specified. The generated random puzzle is guaranteed to have at least one solution at this time, but it might not be unique or guessing might be required.
- If the user requested it, the puzzle is *checked for solvability*, by calling the solver. If the solver doesn't succeed in solving it without recursion, using just its basic deductive mechanisms, the present decimation is discarded and a new one is generated and checked. This saves a lot of generation time at no randomness loss, and eventually succeeds in producing a logically solvable puzzle.
- In either case, the generated puzzle is output, either to the 1-line built-in display (in which case the output format is a no-frills, quite compact variety), or to a real or emulated HP-IL 80-column printer or display, in which case a neat ASCII-graphical representation is used, that is suitable to work it out with pencil and eraser, or copy-paste it to some printing application. Optionally, the full solution can be output at this time as well.

The Coaching features:

As a novel feature, **SUDOGEN** implements *coaching capabilities* as well. This is particularly useful for novices (my wife, say) that do not want the program to fully solve the puzzle for them but only want a hint or two when they get stuck. The provided coaching features work as follows:

- After generation (or input in the case of an externally generated puzzle), the program will ask if the user wants *coaching* and upon acceptance (not asked for directly entered puzzles), it will output *exactly one hint*, which can be any of the following types:

  `[6-3] can only hold 6`

  This means that you can deduce right now that the cell at row 6, column 3, can only hold the value 6, because all other values are already used up in some other cell belonging to its own row, column, or block.

  `[3-4]=only site for 2`

  This means that you can deduce right now that the value 2 can only be placed at row 3, column 4, because all other possible locations for it in the set under

consideration (row, column, or block) also belong to sets that already have a 2 placed somewhere in the set.

`Solved !`

The last hint given already solved the puzzle, so no more hints are necessary and the fully solved grid is output.

`No more forced values`

This can only happen for externally generated puzzles or if the user specified that checking the generated puzzle's solvability wasn't required. The message indicates that the built-in solver couldn't find another forced digit or location, and from now on it will give instead all legal values for each cell in turn, when asked for hints.

`[1-6] admits 579`

This means that the cell at row 1, column 6 can legally hold the values 5, 7, or 9. This kind of hint is given when there are no more forced values that the solver can logically deduct, as stated above.

`No more hints`

All possible hints have already been given, either forced digits/locations or possible legal values for all cells, so no further additional hints are possible. The *partially solved* puzzle is printed which as many filled-in cells as found.

- After being given a hint, the user can try again to solve the puzzle on their own, or else can ask for more hints, until either the puzzle is completely solved, or no more hints are possible. This way, the user can have the minimum help required to allow them to solve the puzzle and learn what can be deducted and when in the process.
- All forced values are hinted at first, *in the proper order to allow them to be logically deducted*. This means that every hinted value is guaranteed to be logically deductible by the user from the present grid state, without blind guessing ever being necessary.
- Once there are no more forced values (which will never happen for generated puzzles which the user specified to be checked for solvability), further hints will be in the form of legal values for each cell, in the proper order too, so that *cells which admit as few values as possible are hinted at first.* So, the order would be cells which only admit 2 legal values, then 3, 4, … and so on. The user can inspect several such hints to try and figure where each digit actually goes.

SUDOGEN Program Listing

Here's the full, commented listing. You don't need to key in comments ("!" lines)

SUDOGEN *(123 lines = 3,902 bytes, 88 lines = 3,267 bytes without comments)*

```
 1 ! *** SUDOGEN: Sudoku Generator & Coach (c) Valentin Albillo, 2006
 2 !
 3 DESTROY ALL @ OPTION BASE 1 @ STD
 4 INTEGER A(9,9),P(9,9),S(9,9),D(9,9),E(9),L(9),Z(81,4) @ DIM B$[41]
 5 !
 6 ! User-defined functions
 7 !
 8 DEF FNR=INT(RND*9)+1
 9 DEF FNW(X)=(X-1) DIV 3
10 DEF FNV=MOD(U+(RND<.5),3)+1+3*FNW(U)
11 !
12 ! Initialization
13 !
14 DISP "Initializing ..." @ FOR I=1 TO 9 @ K=3*FNW(I)+1
15 FOR J=1 TO 9 @ S(I,J)=K+FNW(J) @ NEXT J @ E(I)=2^I @ NEXT I
16 !
17 ! Request user options
18 !
19 INPUT "Gener/Coach G/C? : ","G";R$ @ IF UPRC$(R$)="G" THEN 21
20 DISP "Enter puzzle: " @ MAT INPUT D @ GOTO 53
21 INPUT "Check solve Y/N? : ","Y";R$ @ G=UPRC$(R$)="Y" @ M=62-11*G
22 INPUT "Symmetric   Y/N? : ","N";R$ @ X=UPRC$(R$)="Y"
23 IF G THEN INPUT "# Blanks (1-51)? : ","45";T
24 IF NOT G THEN INPUT "# Blanks (1-62)? : ","51";T
25 IF T<1 OR T>M OR FP(T) THEN 23 ELSE IF X AND MOD(T,4)>1 THEN 22
26 INPUT "Print solut Y/N? : ","N";R$ @ H=UPRC$(R$)="Y"
27 INPUT "Print/Disp  P/D? : ","P";R$ @ Y=UPRC$(R$)="D"
28 !
29 ! Generate puzzle
30 !
31 FOR I=1 TO 9 @ L(I)=I @ NEXT I
32 FOR K=1 TO 20 @ I=FNR @ J=FNR @ U=L(I) @ L(I)=L(J) @ L(J)=U
   @ NEXT K
33 ON MOD(FNR,3)+1 RESTORE 80,81,82 @ READ B$
34 FOR K=0 TO 80 @ I=MOD(K,2)+1
35 A(K DIV 9+1,MOD(K,9)+1)=L(VAL(STR$(NUM(B$[K DIV 2+1])-23)[I,I]))
   @ NEXT K
36 FOR K=1 TO 20
37 U=FNR @ V=FNV @ FOR I=1 TO 9 @ N=A(I,U) @ A(I,U)=A(I,V) @ A(I,V)=N
   @ NEXT I
38 U=FNR @ V=FNV @ FOR J=1 TO 9 @ N=A(U,J) @ A(U,J)=A(V,J) @ A(V,J)=N
   @ NEXT J
39 NEXT K @ W=0 @ IF RND<.5 THEN MAT A=TRN(A)
40 W=W+1 @ DISP " Gen ";STR$(W);":"; @ MAT D=A
41 IF X THEN GOSUB 64 ELSE GOSUB 60
42 IF G THEN DISP " chk "; ELSE DISP @ MAT P=A @ GOTO 48
43 !
44 ! Test the puzzle's solvability and output if solvable
45 !
46 MAT P=D @ F=0
47 CALL TRY(P,F,0,S,E,Z,K) @ IF F#2 THEN DISP " NOK" @ GOTO 40
   ELSE DISP " OK!"
```

```
 48 CALL SDP(D,Y) @ IF H THEN DISP "Solution:" @ CALL SDP(P,Y)
 49 !
 50 ! Offer coaching and if yes, find and output hints
 51 !
 52 INPUT "Coach Y/N? : ","Y";R$ @ IF UPRC$(R$)#"Y" THEN END
 53 CALL TRY(D,F,1,S,E,Z,K) @ IF F<2 THEN GOSUB 69 @ GOTO 53
 54 IF F=2 THEN DISP "Solved!" ELSE IF F=3 THEN DISP "Illegal puzzle"
 55 IF F=4 THEN GOSUB 71 @ DISP "No more hints"
 56 CALL SDP(D,Y) @ END
 57 !
 58 ! Subroutines: blank, symmetric blank, wait key, output hints
 59 !
 60 FOR K=1 TO T
 61 I=FNR @ J=FNR @ IF D(I,J) THEN D(I,J)=0 ELSE 61
 62 NEXT K @ RETURN
 63 !
 64 IF MOD(T,2) THEN D(5,5)=0
 65 FOR K=1 TO T DIV 4
 66 I=FNR @ J=FNR @ IF I=5 OR J=5 OR NOT D(I,J) THEN 66
 67 D(I,J)=0 @ D(I,10-J)=0 @ D(10-I,J)=0 @ D(10-I,10-J)=0 @ NEXT K
    @ RETURN
 68 !
 69 R$=KEY$ @ IF R$="" THEN 69 ELSE IF R$="#38" THEN RETURN ELSE END
 70 !
 71 DISP "No more forced values" @ FOR N=2 TO 9 @ FOR U=1 TO K
 72 IF Z(U,3)#N THEN 76 ELSE I=Z(U,1) @ J=Z(U,2) @ W=Z(U,4)
 73 DISP "[";STR$(I);"-";STR$(J);"] admits ";
 74 FOR V=1 TO 9 @ IF NOT BIT(W,V) THEN DISP STR$(V);
 75 NEXT V @ DISP @ GOSUB 69
 76 NEXT U @ NEXT N @ RETURN
 77 !
 78 ! Data for base puzzle classes
 79 !
 80 DATA "Xi^Bs$pFOFtX)4/Xj(O^Bso6scD)P;4FAJyV>WiP!"
 81 DATA "[,>Db@Z3tx9Q#Z*K;kBQTx-[u`&&Uk`tu^K[i`wM."
 82 DATA "7*Mck\i`JQ^;H1Q/r`u;^iP#=OH*P[.G;H0(2D$yS"
 83 !
 84 ! Subprogram to forcibly solve puzzles and/or generate hints
 85 !
 86 SUB TRY(P(,),F,Z,S(,),E(),A(,),K)
 87 INTEGER N1(9,9),N2(9,9),N3(9,9),G(9,9),R(9),C(9),B(9)
 88 M=0 @ FOR I=1 TO 9 @ FOR J=1 TO 9 @ IF P(I,J) THEN D=E(P(I,J))
    @ GOSUB 113
 89 NEXT J @ NEXT I
 90 F=2 @ K=0 @ FOR I=1 TO 9 @ U=R(I) @ FOR J=1 TO 9 @ V=C(J)
    @ IF P(I,J) THEN 98
 91 K=K+1 @ H=S(I,J) @ F=F*(F#2) @ L=BINIOR(BINIOR(U,V),B(H))
 92 IF L=1022 THEN F=3 @ END
 93 D=BINAND(BINCMP(L),1023)-1 @ N=BIT(D,1)+BIT(D,2)+BIT(D,3)
 94 N=N+BIT(D,4)+BIT(D,5)+BIT(D,6)+BIT(D,7)+BIT(D,8)+BIT(D,9)
 95 IF N#1 THEN A(K,1)=I @ A(K,2)=J @ A(K,3)=N @ A(K,4)=L @ GOTO 98
 96 P(I,J)=LOG2(D) @ F=1 @ M=M+1 @ R(I)=R(I)+D @ C(J)=C(J)+D
    @ B(H)=B(H)+D
 97 IF Z THEN GOSUB 110 @ DISP " can only hold";P(I,J) @ END
 98 NEXT J @ NEXT I @ IF NOT Z THEN DISP ".";
 99 IF F=1 THEN 90 ELSE IF F=2 THEN END
100 MAT N1=ZER @ MAT N2=ZER @ MAT N3=ZER @ N=0 @ FOR U=1 TO K
    @ I=A(U,1)
```

```
101 J=A(U,2) @ H=S(I,J) @ L=A(U,4) @ FOR V=1 TO 9 @ IF BIT(L,V)
    THEN 105
102 IF N1(I,V) THEN N1(I,V)=-1 ELSE N1(I,V)=J
103 IF N2(J,V) THEN N2(J,V)=-1 ELSE N2(J,V)=I
104 IF N3(H,V) THEN N3(H,V)=-1 ELSE N3(H,V)=I @ G(H,V)=J
105 NEXT V @ NEXT U @ FOR U=1 TO 9 @ FOR V=1 TO 9
106 J=N1(U,V) @ IF J>0 THEN I=U @ GOSUB 111
107 I=N2(U,V) @ IF I>0 THEN J=U @ GOSUB 111
108 I=N3(U,V) @ IF I>0 THEN J=G(U,V) @ GOSUB 111
109 NEXT V @ NEXT U @ IF N THEN 90 ELSE F=4 @ END
110 DISP "[";STR$(I);"-";STR$(J);"]"; @ RETURN
111 IF P(I,J) THEN RETURN ELSE M=M+1 @ N=1 @ P(I,J)=V @ D=E(V)
112 IF Z THEN GOSUB 110 @ DISP "=only site for";V @ END
113 H=S(I,J) @ R(I)=R(I)+D @ C(J)=C(J)+D @ B(H)=B(H)+D @ RETURN
114 !
115 ! Subprogram to pretty-print or display puzzles
116 !
117 SUB SDP(A(,),Y) @ IF Y THEN MAT DISP USING "DX";A; @ END
118 DIM A$(2)[37] @ A$(2)="+--+--+--+--+--+--+--+--+--+"
119 C$=CHR$(124) @ A$(1)="+        +        +        +"
120 DISP @ DISP A$(2) @ FOR I=1 TO 9 @ FOR J=1 TO 9
121 IF MOD(J-1,3) THEN DISP " "; ELSE DISP C$;
122 IF A(I,J) THEN DISP " "&STR$(A(I,J)); ELSE DISP " .";
123 NEXT J @ DISP C$ @ DISP A$(NOT MOD(I,3)+1) @ NEXT I
```

Notes:

- The program requires an additional 2,500 bytes of free RAM available for its variables, etc. It also makes use of keywords from the **Math ROM** and **HP-IL ROM**. **Emu71** executes it *200-300X faster* on a P4 @ 2.4 Ghz PC.
- When specifying that the generated puzzle is to be checked for solvability (unique solution, no guessing necessary), the average number of tries the generator needs to perform until it finds a suitable one (and thus the running time) heavily depends on the number of blank cells specified, as follows:

Blank cells	Average # of tries	Blank cells	Average # of tries
<=25	1	**40**	3
30	1-2	**45**	10
35	2	**50**	>=75

  So it's advisable to either refrain from specifying more than 45 blank cells in the generated puzzle in order to avoid excessive running times, or else avoid checking for solvability if generating puzzles with more than 45 blanks.
- For the sake of completeness, here's a brief explanation of ***Sudoku*** puzzles: you're given a 9x9 grid, each cell to be occupied by a *single* digit 1-9, such that *each* of the 9 columns, rows, and 3x3 non-overlapping blocks contain all digits without repetition. Initially some cells are already filled-in and you must fill the rest.

Usage instructions

- To start the program, press or type:

  `[RUN] -> Initializing ...`

- Answer the prompts requesting which options do you want:

  `Print/Disp  P/D? : P`

  If you select **P**, the generated puzzle and optionally its solution will be *pretty-printed*, using ASCII chars for the grid, in a format suitable for actual paper copy for the user to solve with pencil and eraser, see examples.

  If you select **D**, they will be output in a *compact* format suitable for the built-in 1-line LCD display.

  `Gener/Coach G/C? : G`

  If you select **G**, the program will generate and print a puzzle, then it will ask if you want *coaching*.

  If you select **C**, the program will allow you to enter an *externally generated puzzle* (e.g., from a newspaper) and will then proceed directly to coaching, outputting the first hint, see **Coaching** below. The puzzle is entered at this prompt:

  `Enter puzzle: D(1,1)?`

  and you must then enter the values for *all* 81 cells separated by commas, row by row from top to bottom and from left to right, where a **0** must be entered for blank cells. See **Examples** below.

  `Check solve Y/N? : Y`

  If you select **Y**, the program will make sure the generated puzzle has a *unique* solution and that the full solution *can be found by logical deductions* alone, without guessing being ever necessary. Also, later coaching will succeed in giving hints that can go on till the puzzle is completely solved. This may require several tries before a suitable puzzle is found, their number depending on the number of blank cells specified. See **Notes** above.

  If you select **N**, the generated puzzle *is still guaranteed to be solvable*, but the solution may *not* be unique and may require *guessing* to fully complete. Also, later coaching may not be able to give hints that completely solve the puzzle, see **Coaching** below. On the other hand, the puzzle will be generated *very quickly*, regardless of the number of blanks specified, see **Notes** above.

  `Symmetric   Y/N? : N`

  If you select **Y**, a symmetric puzzle will be generated. If you select N, the puzzle will most likely be non-symmetric.

  `# Blanks (1-51)? : 45`

  You must enter the number of blank cells you want in the puzzle, within the specified limits. The greater the number of blank cells, the more difficult the

puzzle will be and the more it will take to generate if solvability checking has been specified. The legal values are as follows:

Solvability checked: from 1 to 51 blanks (<=45 recommended)

Solvability unchecked: from 1 to 62 blanks

Additionally, if Symmetric puzzle generation is specified, the number of blanks *must be a multiple of 4 or a multiple of 4 plus 1*. For example, 40 and 41 would be legal values, but entering 42 would be illegal and you would be prompted again if you really want a Symmetric puzzle.

```
Print solut Y/N? : N
```

If you select **Y**, the solution will be printed/displayed after the puzzle itself, whether you specified to check for solvability or not.

If you select **N**, the solution won't be printed unless you later specify that you want coaching and the puzzle is fully solvable, which will always be the case if you specified to check for solvability.

- Once all prompts have been answered, the program will proceed to generate the puzzle as specified (unless an external puzzle has been entered, in which case it will directly enter coaching, see below). If checking solvability was specified, you'll see messages while each tentative puzzle is generated and checked:

```
Gen 1: chk ..... NOK
Gen 2: chk ..... NOK
Gen 3: chk ................. OK!
```

Once a suitable puzzle is found, the generation is over. If checking for solvability was not specified, the very *first* try will succeed at once.

- The generated puzzle will then be output, either in pretty-print or compact format, and the solution will be printed as well if specified.

Coaching

- Once the puzzle has been generated and output you'll be prompted if you want coaching (if directly entered, you're not prompted but enter coaching at once):

Coach Y/N? : Y

If you select **N**, the program will end. If you select **Y**, the coaching features will be activated, see **The Coaching features** above.

The hints supplied on an individual basis by the coaching process can help the user to solve the puzzle after getting stuck, and can also be used to learn and understand the solving procedure. Upon being given a forced value for a cell, the user should try and discover why the coach says this value is forced in the present status of the grid. A little study of what is it that makes that value forced will undoubtely result in the beginner quickly sharpening their deductive skills.

Assorted examples

1. Generate, check solvability, non-symmetric, 45 blanks

```
>RANDOMIZE 1 [ENTER]
>RUN         [ENTER]
Initializing ...
Print/Disp  P/D? : P     [ENTER]
Gener/Coach G/C? : G     [ENTER]
Check solve Y/N? : Y     [ENTER]
Symmetric   Y/N? : N     [ENTER]
# Blanks (1-51)? : 45    [ENTER]
Print solut Y/N? : N     [ENTER]
 Gen 1: chk ....... NOK
 Gen 2: chk ........ NOK
 Gen 3: chk ................. OK!

{it took only three tries to find a
 suitable puzzle guaranteed to have
 a unique solution which can be found
 by deduction alone, without guessing}


Coach Y/N? : Y           [ENTER]
[6-3] can only hold 6    [ENTER]
[6-4] can only hold 7    [ENTER]
[6-8] can only hold 9    [ENTER]
[3-2]=only site for 4    [ENTER]
[3-4]=only site for 2    [ENTER]
[3-9]=only site for 1    [ENTER]
[7-9] can only hold 6    [ENTER]
[4-9] can only hold 5    [ENTER]
       ...
[5-8] can only hold 8    [ENTER]
[5-9] can only hold 7    [ENTER]
Solved!

{as the puzzle was guaranteed to be
 solvable by deduction alone, you
 get precise hints one by one till
 it´s completely solved}
```

```
+--+--+--+--+--+--+--+--+--+
| .  2  8| 1  4  .| .  6  .|
+        +        +        +
| .  9  .| .  3  .| 2  .  4|
+        +        +        +
| 6  .  3| .  7  .| .  5  .|
+--+--+--+--+--+--+--+--+--+
| .  3  7| 9  .  .| .  4  .|
+        +        +        +
| .  .  9| .  .  .| .  .  .|
+        +        +        +
| 4  8  .| .  5  3| 1  .  2|
+--+--+--+--+--+--+--+--+--+
| 8  7  .| .  9  2| .  3  .|
+        +        +        +
| .  .  .| .  6  .| .  2  8|
+        +        +        +
| .  6  2| .  8  .| 5  .  .|
+--+--+--+--+--+--+--+--+--+
| 7  2  8| 1  4  5| 9  6  3|
+        +        +        +
| 1  9  5| 8  3  6| 2  7  4|
+        +        +        +
| 6  4  3| 2  7  9| 8  5  1|
+--+--+--+--+--+--+--+--+--+
| 2  3  7| 9  1  8| 6  4  5|
+        +        +        +
| 5  1  9| 6  2  4| 3  8  7|
+        +        +
| 4  8  6| 7  5  3| 1  9  2|
+--+--+--+--+--+--+--+--+--+
| 8  7  1| 5  9  2| 4  3  6|
+        +        +        +
| 9  5  4| 3  6  1| 7  2  8|
+        +        +        +
| 3  6  2| 4  8  7| 5  1  9|
+--+--+--+--+--+--+--+--+--+
```

2. Generate, check solvability, symmetric, 45 blanks

```
>RANDOMIZE 4
>RUN
Initializing ...
Print/Disp  P/D? : P
Gener/Coach G/C? : G
Check solve Y/N? : Y
Symmetric   Y/N? : Y
# Blanks (1-51)? : 45
Print solut Y/N? : N
 Gen 1: chk .... NOK
 Gen 2: chk ...... OK!

{this time it took only two tries to
 find a uniquely solvable puzzle, which
 has 45 blank cells for the user to
 fill up, plus it's symmetric to boot}
```

```
+--+--+--+--+--+--+--+--+--+
| .  5  .| 4  7  9| .  6  .|
+        +        +        +
| 9  .  .| .  2  .| .  .  8|
+        +        +        +
| .  1  .| .  6  .| .  9  .|
+--+--+--+--+--+--+--+--+--+
| .  .  7| .  9  .| 6  .  .|
+        +        +        +
| 6  4  9| 1  .  7| 8  2  5|
+        +        +        +
| .  .  3| .  8  .| 7  .  .|
+--+--+--+--+--+--+--+--+--+
| .  9  .| .  1  .| .  8  .|
+        +        +        +
| 2  .  .| .  4  .| .  .  3|
+        +        +        +
| .  7  .| 9  5  2| .  4  .|
+--+--+--+--+--+--+--+--+--+
```

```
Coach Y/N? : Y
[5-5] can only hold 3
[6-2] can only hold 2
[4-2] can only hold 8
[6-8] can only hold 1
[4-8] can only hold 3
[4-9] can only hold 4
[4-6] can only hold 5
[4-1] can only hold 1
[4-4] can only hold 2
      ...
[9-7] can only hold 1
[9-9] can only hold 6
Solved!

{the coaching succeeds once more in
 giving all necessary hints to fully
 solve the puzzle}
```

```
+--+--+--+--+--+--+--+--+--+
| 8  5  2| 4  7  9| 3  6  1|
+        +        +        +
| 9  3  6| 5  2  1| 4  7  8|
+        +        +        +
| 7  1  4| 8  6  3| 5  9  2|
+--+--+--+--+--+--+--+--+--+
| 1  8  7| 2  9  5| 6  3  4|
+        +        +        +
| 6  4  9| 1  3  7| 8  2  5|
+        +        +        +
| 5  2  3| 6  8  4| 7  1  9|
+--+--+--+--+--+--+--+--+--+
| 4  9  5| 3  1  6| 2  8  7|
+        +        +        +
| 2  6  1| 7  4  8| 9  5  3|
+        +        +        +
| 3  7  8| 9  5  2| 1  4  6|
+--+--+--+--+--+--+--+--+--+
```

3. Generate, do not check solvability, non-symmetric, 56 blanks

```
>RANDOMIZE 7
>RUN
Initializing ...
Print/Disp  P/D? : P
Gener/Coach G/C? : G
Check solve Y/N? : N
Symmetric   Y/N? : N
# Blanks (1-62)? : 56
Print solut Y/N? : N
 Gen 1:

{as the number of blanks is >45 we don't
 check solvability this time, to avoid
 potentially excessive generation times.
 The generated puzzle is still guaranteed
 to be solvable, but the solution might
 not be unique, and guessing might be
 necessary in order to solve it}

Coach Y/N? : Y
[1-8]=only site for 5
[6-8] can only hold 9
[5-9] can only hold 1
[4-7] can only hold 2
[6-7] can only hold 6
[6-9] can only hold 5
[5-4]=only site for 3
No more forced values
[2-7] admits 89
[3-7] admits 89
[5-1] admits 89
[5-2] admits 89
[7-8] admits 37
[7-9] admits 89
[8-9] admits 89
[9-7] admits 14
[9-8] admits 37
   ...
[7-1] admits 1234578
[7-5] admits 1234579    {the built-in solver was able to forcibly fill up
No more hints            7 cells and gives legal values for the rest}
```

```
+--+--+--+--+--+--+--+--+--+
| .  .  .| .  .  .| 3  .  4|
+        +        +        +
| .  .  .| .  .  .| .  6  2|
+        +        +        +
| .  .  2| 5  .  .| .  1  7|
+--+--+--+--+--+--+--+--+--+
| 6  .  4| .  .  .| .  8  3|
+        +        +        +
| .  .  5| .  6  2| 7  4  .|
+        +        +        +
| .  2  3| .  .  .| .  .  .|
+--+--+--+--+--+--+--+--+--+
| .  .  6| .  .  .| .  .  .|
+        +        +        +
| .  .  .| .  .  .| 5  2  .|
+        +        +        +
| .  .  9| 8  .  .| .  .  6|
+--+--+--+--+--+--+--+--+--+
+--+--+--+--+--+--+--+--+--+
| .  .  .| .  .  .| 3  5  4|
+        +        +        +
| .  .  .| .  .  .| .  6  2|
+        +        +        +
| .  .  2| 5  .  .| .  1  7|
+--+--+--+--+--+--+--+--+--+
| 6  .  4| .  .  .| 2  8  3|
+        +        +        +
| .  .  5| 3  6  2| 7  4  1|
+        +        +        +
| .  2  3| .  .  .| 6  9  5|
+--+--+--+--+--+--+--+--+--+
| .  .  6| .  .  .| .  .  .|
+        +        +        +
| .  .  .| .  .  .| 5  2  .|
+        +        +        +
| .  .  9| 8  .  .| .  .  6|
+--+--+--+--+--+--+--+--+--+
```

4. Generate, do not check solvability, symmetric, 49 blanks

```
>RANDOMIZE 5
>RUN
Initializing ...
Print/Disp  P/D? : P
Gener/Coach G/C? : G
Check solve Y/N? : N
Symmetric   Y/N? : Y
# Blanks (1-62)? : 49
Print solut Y/N? : N
 Gen 1:

{again we do not check solvability, as
 the desired number of blank cells
 exceeds 45, but this time we request
 that the generated puzzle be symmetric.
 As before, the puzzle is solvable, but
 might have more than one solution and
 guessing might be necessary.

Coach Y/N? : Y
[5-5] can only hold 4
[7-8] can only hold 4
[7-7] can only hold 9
[2-1]=only site for 9
[2-3]=only site for 5
[1-1]=only site for 7
[4-9]=only site for 9
[8-6]=only site for 9
No more forced values
[1-6] admits 26
[1-9] admits 28
[2-6] admits 46
[3-8] admits 12
[7-9] admits 35
[8-3] admits 34
[8-4] admits 34
[8-7] admits 27
[9-9] admits 23
    ...
[6-3] admits 34678      {this time the coach offered 8 forced-value hints
[6-9] admits 12478       and gave legal-value hints for the remaining cells}
No more hints
```

```
+--+--+--+--+--+--+--+--+--+
| .  4  1| .  3  .| 5  9  .|
+        +        +        +
| .  2  .| .  1  .| .  3  .|
+        +        +        +
| .  .  .| 9  7  5| .  .  .|
+--+--+--+--+--+--+--+--+--+
| .  .  .| .  6  .| .  .  .|
+        +        +        +
| 1  9  2| 7  .  8| 3  5  6|
+        +        +        +
| .  .  .| .  9  .| .  .  .|
+--+--+--+--+--+--+--+--+--+
| .  .  .| 1  2  7| .  .  .|
+        +        +        +
| .  1  .| .  8  .| .  6  .|
+        +        +        +
| .  7  9| .  5  .| 1  8  .|
+--+--+--+--+--+--+--+--+--+
+--+--+--+--+--+--+--+--+--+
| 7  4  1| .  3  .| 5  9  .|
+        +        +        +
| 9  2  5| .  1  .| .  3  .|
+        +        +        +
| .  .  .| 9  7  5| .  .  .|
+--+--+--+--+--+--+--+--+--+
| .  .  .| .  6  .| .  .  9|
+        +        +        +
| 1  9  2| 7  4  8| 3  5  6|
+        +        +        +
| .  .  .| .  9  .| .  .  .|
+--+--+--+--+--+--+--+--+--+
| .  .  .| 1  2  7| 9  4  .|
+        +        +        +
| .  1  .| .  8  9| .  6  .|
+        +        +        +
| .  7  9| .  5  .| 1  8  .|
+--+--+--+--+--+--+--+--+--+
```

5. Coach given external puzzle (fully solvable puzzle)

				5	4		9	
2	4	6	9		3		1	
				6	1	2		
4	8				6		5	
	5	7			9			
		3			8			9
	3	2				9	7	6
			6	9	2	4	3	
			8		7	1	2	

This time we're not generating a puzzle ourselves but just want to make use of the coaching features while solving an externally generated puzzle (say taken from a newspaper).

We simply select <u>Coach</u> instead of <u>Generate</u>, and as you will see, in this case the built-in solver/coach can completely solve it, so it offers forced-value hints throughout until it is fully solved (or the user doesn't ask for any further hints but decides to go on their own without further assistance).

```
>RUN
Initializing ...
Print/Disp  P/D? : P
Gener/Coach G/C? : C
Enter puzzle:
D(1,1)? 0,0,0,0,5,4,0,9,0,2,4,6,9,0,
        3,0,1,0,0,0,0,0,6,1,2,0,0
D(4,1)? 4,8,0,0,0,6,0,5,0,0,5,7,0,0,
        9,0,0,0,0,0,3,0,0,8,0,0,9
D(7,1)? 0,3,2,0,0,0,9,7,6,0,0,0,6,9,
        2,4,3,0,0,0,0,8,0,7,1,2,0
[3-4] can only hold 7
[1-4] can only hold 2
[2-5] can only hold 8
[3-2] can only hold 9
[7-6] can only hold 5
      ...
[9-3] can only hold 4
[9-5] can only hold 3
Solved!
```

```
+--+--+--+--+--+--+--+--+--+
| 7  1  8| 2  5  4| 6  9  3|
+        +        +        +
| 2  4  6| 9  8  3| 5  1  7|
+        +        +        +
| 3  9  5| 7  6  1| 2  8  4|
+--+--+--+--+--+--+--+--+--+
| 4  8  9| 1  7  6| 3  5  2|
+        +        +        +
| 6  5  7| 3  2  9| 8  4  1|
+        +        +        +
| 1  2  3| 5  4  8| 7  6  9|
+--+--+--+--+--+--+--+--+--+
| 8  3  2| 4  1  5| 9  7  6|
+        +        +        +
| 5  7  1| 6  9  2| 4  3  8|
+        +        +        +
| 9  6  4| 8  3  7| 1  2  5|
+--+--+--+--+--+--+--+--+--+
```

6. Coach given external puzzle (not fully solvable)

	4			2			8	
				8				
7	8		4	3	1		5	2
				9				
9	2	1	3		8	5	7	4
				4				
8	3		2	7	6		4	5
				1				
	1			5			6	

Now the externally generated puzzle we want coaching for can't be fully solved by the built-in solver without using recursion, i.e. by its deductive mechanisms alone.
Instead, the solver/coach gives as many forced-value hints as it can deductively find, 7 in all, then offers hints consisting of the legal values for all remaining cells, starting with the ones which admit the fewest possible legal values. The user is then expected to make their own deductions based on this useful and comprehensive info provided by the coach. Finally, the partially solved grid is output.

```
>RUN
Initializing ...
Print/Disp  P/D? : P
Gener/Coach G/C? : C
Enter puzzle:
D(1,1)? 0,4,0,0,2,0,0,8,0,0,0,0,0,8,
        0,0,0,0,7,8,0,4,3,1,0,5,2
D(4,1)? 0,0,0,0,9,0,0,0,0,9,2,1,3,0,
        8,5,7,4,0,0,0,0,4,0,0,0,0
D(7,1)? 8,3,0,2,7,6,0,4,5,0,0,0,0,1,
        0,0,0,0,0,1,0,0,5,0,0,6,0
[5-5] can only hold 6
[7-3] can only hold 9
[3-3] can only hold 6
[3-7] can only hold 9
[7-7] can only hold 1
[2-2]=only site for 9
[2-7]=only site for 4
No more forced values
[1-3] admits 35
[2-6] admits 57
      ...
[6-9] admits 13689
No more hints
```

```
+--+--+--+--+--+--+--+--+--+
| .  4  .| .  2  .| .  8  .|
+        +        +        +
| .  9  .| .  8  .| 4  .  .|
+        +        +        +
| 7  8  6| 4  3  1| 9  5  2|
+--+--+--+--+--+--+--+--+--+
| .  .  .| .  9  .| .  .  .|
+        +        +        +
| 9  2  1| 3  6  8| 5  7  4|
+        +        +        +
| .  .  .| .  4  .| .  .  .|
+--+--+--+--+--+--+--+--+--+
| 8  3  9| 2  7  6| 1  4  5|
+        +        +        +
| .  .  .| .  1  .| .  .  .|
+        +        +        +
| .  1  .| .  5  .| .  6  .|
+--+--+--+--+--+--+--+--+--+
```