\*&---------------------------------------------------------------------\*

\*& Report ZTEST1

\*&

\*&---------------------------------------------------------------------\*

\*& Desc: Sudoku Solver and Generator

\*& Developed by: Manohar Potnuru

\*&---------------------------------------------------------------------\*

REPORT ztest1.

\* main table to process the sudoku values

TYPES: BEGIN OF ty\_sud,

con TYPE string, "concatenate row and col

row TYPE c, "row

col TYPE c, "column

val TYPE c, "Value

ava TYPE i, "available

blk TYPE i, "Block

seq TYPE i, "Sequence

nsq TYPE i, " New seq.

tnt TYPE c, "Tried, Not Tried

pnp TYPE c, "Possible , Not possible

END OF ty\_sud.

\* input excel file for upload/download

TYPES: BEGIN OF ty\_file,

c1 TYPE string,

c2 TYPE string,

c3 TYPE string,

c4 TYPE string,

c5 TYPE string,

c6 TYPE string,

c7 TYPE string,

c8 TYPE string,

c9 TYPE string,

r TYPE string,

END OF ty\_file.

DATA: gt\_sud TYPE STANDARD TABLE OF ty\_sud,

gt\_sudiu TYPE STANDARD TABLE OF ty\_sud, "Initial Upload

gt\_sudf TYPE STANDARD TABLE OF ty\_sud,

gt\_sud1 TYPE STANDARD TABLE OF ty\_sud,

gs\_sud TYPE ty\_sud,

gs\_sud1 TYPE ty\_sud,

gs\_sud2 TYPE ty\_sud,

gt\_file TYPE STANDARD TABLE OF ty\_file,

gs\_file1 TYPE ty\_file,

gs\_file2 TYPE ty\_file,

gs\_file3 TYPE ty\_file,

gs\_file4 TYPE ty\_file,

gs\_file5 TYPE ty\_file,

gs\_file6 TYPE ty\_file,

gs\_file7 TYPE ty\_file,

gs\_file8 TYPE ty\_file,

gs\_file9 TYPE ty\_file,

gs\_filec TYPE ty\_file.

DATA: random TYPE i,

rand TYPE c,

row TYPE c,

col TYPE c,

ava TYPE c,

gap TYPE i,

seq TYPE i,

nsq TYPE i,

chk TYPE c,

con TYPE string,

index TYPE sy-tabix,

lines TYPE i,

unsolved TYPE i,

unsolvable TYPE c,

given TYPE i,

unprocessed TYPE i,

restart TYPE i.

DATA: w\_text(80),

count TYPE string,

tcount TYPE string.

DATA: itab\_upload\_file TYPE alsmex\_tabline OCCURS 0 WITH HEADER LINE,

itab\_upload\_file\_temp TYPE alsmex\_tabline OCCURS 0 WITH HEADER LINE.

\*---------------------------------------------------------------------\*

\* SELECTION SCREEN

\*---------------------------------------------------------------------\*

SELECTION-SCREEN BEGIN OF BLOCK b2 WITH FRAME TITLE TEXT-002.

PARAMETERS : p\_disp RADIOBUTTON GROUP r1 DEFAULT 'X' , "random sud print on screen

p\_down RADIOBUTTON GROUP r1 , "random sud download

p\_upl RADIOBUTTON GROUP r1 . "uplaod sud and its result on screen

SELECTION-SCREEN END OF BLOCK b2.

SELECTION-SCREEN BEGIN OF BLOCK b3 WITH FRAME TITLE TEXT-001.

PARAMETERS: p\_file TYPE rlgrap-filename MODIF ID m1 . " File Path

SELECTION-SCREEN END OF BLOCK b3.

\*---------------------------------------------------------------------\*

\* AT SELECTION SCREEN

\*---------------------------------------------------------------------\*

AT SELECTION-SCREEN.

IF p\_disp NE 'X' AND p\_file IS INITIAL.

SET CURSOR FIELD 'P\_FILE'.

MESSAGE 'Enter file path' TYPE 'E' DISPLAY LIKE 'S'.

ENDIF.

\*---------------------------------------------------------------------\*

\* AT SELECTION SCREEN

\*---------------------------------------------------------------------\*

AT SELECTION-SCREEN ON VALUE-REQUEST FOR p\_file.

\* Opening window for path selection

PERFORM get\_filename.

\*---------------------------------------------------------------------\*

\* START-OF-SELECTION

\*---------------------------------------------------------------------\*

START-OF-SELECTION.

\* Fill all possible 9 values in 81 cells.

PERFORM fill\_initial\_possible.

IF p\_upl EQ 'X'. "If upload an initial sudoku via an excel file

\* Upload Excel data into internal table structure of Function module

PERFORM upload\_excel.

PERFORM fill\_from\_upload.

" check for duplicacy from the initial upload.

PERFORM initial\_check CHANGING chk.

IF chk = 'X'.

FREE: gt\_sud[].

unsolvable = 'X'.

ENDIF.

ELSE.

\* Fill the three diagonal blocks(3 x 3) first with random numbers

\* because these blocks can be filled independently,

SORT gt\_sud BY seq blk row col val ava.

\* for seq 1.

PERFORM fill\_diagonal\_blocks.

ENDIF.

PERFORM fill\_new\_sequence.

\* Do recursive fill- This is key aspect of solving a sudoku.

\* This perform is called recursively as and when needed.

\* This same perform is called with in this perform,

\*so it is named it as recursive\_fill

PERFORM recursive\_fill CHANGING chk.

\*---------------------------------------------------------------------\*

\* End-OF-SELECTION

\*---------------------------------------------------------------------\*

END-OF-SELECTION.

gt\_sud[] = gt\_sudf[].

SORT gt\_sud BY row col val .

PERFORM final\_check CHANGING chk. " check for duplicacy

IF p\_upl NE 'X'.

\* Now we have with us a unique Sudoku, Now we need to clear few values

\* randomly such that the player can tried solving on his own

PERFORM clear\_few\_values.

ENDIF.

IF p\_down NE 'X'.

\* display the random sudoku generated as well as the uploaded sudoku

\* if any and the sovled sudoku( for a freshly uploaded sukoku from a file)

PERFORM display\_sud.

ELSE.

\* Download the generated sudoku file.

PERFORM write\_sud\_file.

ENDIF.

\*&---------------------------------------------------------------------\*

\*& Form recursive\_fill

\*&---------------------------------------------------------------------\*

FORM recursive\_fill CHANGING chk.

\* no. of recursion count

count = count + 1.

tcount = tcount + 1.

IF count = 2000.

CLEAR count.

\* Break-point

ENDIF.

CONCATENATE 'Recursion number:' count INTO w\_text SEPARATED BY space.

CALL FUNCTION 'SAPGUI\_PROGRESS\_INDICATOR'

EXPORTING

text = w\_text.

DATA: lv\_nsq TYPE i.

CLEAR chk.

\* restart indicator is used to process the internal table from the

\* position when this recursive perform is called

IF restart = 0 .

restart = 1.

ENDIF.

\* unprocessed number is the already fixed value from the initial

\* upload or randomly generated diagonal block cell,

\* because we have sorted the internal table such that

\* already fixed cells are palced at last

LOOP AT gt\_sud INTO gs\_sud FROM restart.

IF sy-tabix = unprocessed.

\* see if 81 cells have been filled appropriately

PERFORM map\_complete\_possibility CHANGING chk.

IF chk = 'X'.

\* Do a recursive fill, here restart is set to 0.

PERFORM recursive\_fill CHANGING chk.

ENDIF.

ENDIF.

\* Further douwn the processing, there is a chance that if the sudoku

\*is not solvable, then the main internal table is freed.

IF gt\_sud[] IS INITIAL.

EXIT.

ENDIF.

\* check for the row with not yet sovled, not yet tried cell

CHECK gs\_sud-val = '' AND gs\_sud-pnp = '' AND gs\_sud-tnt = ''.

\* check for duplicacy in rows, columns and blocks for the Tried cells

PERFORM check\_in\_sud CHANGING chk.

IF chk = 'X'.

CLEAR chk.

CONTINUE.

ENDIF.

\* nsq ranging from 1 to 81 as per the latest cell processed/Tried

\*and gs\_sud-nsq is the not tried cell

lv\_nsq = gs\_sud-nsq - nsq.

\* difference greater than 1 implies all tried possiblities

\* are exhausted for that particular cell

IF lv\_nsq GT 1.

nsq = 0.

DO unsolved TIMES.

nsq = nsq + 1.

LOOP AT gt\_sud INTO gs\_sud2 WHERE nsq = nsq

AND pnp NE 'N'.

\* exit at the point where there can be tried a different

\* possible/available value for that particular cell

EXIT.

ENDLOOP.

IF sy-subrc IS NOT INITIAL.

EXIT.

ENDIF.

ENDDO.

DO.

IF nsq LE 0.

chk = 'X'.

unsolvable = 'X'.

FREE: gt\_sud[].

EXIT.

ENDIF.

\* here 'S' is called suppressed, because in the previous iterations

\* we would have chosed a particular value for that cell and

\* would have suppressed the remaining alternatives.

\* Search for that particualar cell sequence

READ TABLE gt\_sud WITH KEY nsq = nsq

pnp = 'S' TRANSPORTING NO FIELDS.

IF sy-subrc IS NOT INITIAL.

nsq = nsq - 1.

CONTINUE.

ELSE.

EXIT.

ENDIF.

ENDDO.

\* try from this cell assuming that cells processed before this

\* cell are derived correctly to become a unique sudoku

READ TABLE gt\_sud INTO gs\_sud2 WITH KEY nsq = nsq tnt = 'T' pnp = 'P'.

IF sy-subrc IS INITIAL.

\* here mark this available value to not possible and unsuppress the

\* already suppressed value for this particular cell

\* also unmark the not possible values from other cells and columns

\* because we have concluded that the previously tried option was wrongly picked

PERFORM uncheck\_in\_sud CHANGING chk.

READ TABLE gt\_sud WITH KEY nsq = nsq TRANSPORTING NO FIELDS.

IF sy-subrc IS INITIAL.

restart = sy-tabix.

\* mark all the suppressed option to unsppressed option

LOOP AT gt\_sud INTO gs\_sud2 FROM restart WHERE val = '' AND pnp = 'S'.

IF gs\_sud2-val NE ''.

EXIT.

ENDIF.

gs\_sud2-pnp = ''.

gs\_sud2-tnt = ''.

MODIFY gt\_sud FROM gs\_sud2 INDEX sy-tabix TRANSPORTING pnp tnt.

CLEAR gs\_sud2.

ENDLOOP.

ENDIF.

ENDIF.

\* Do a recursive fill with the derived resta4rt value

PERFORM recursive\_fill CHANGING chk.

ENDIF.

IF gt\_sud[] IS INITIAL.

EXIT.

ENDIF.

nsq = gs\_sud-nsq .

gs\_sud-pnp = 'P'. " Mark possible

gs\_sud-tnt = 'T'. " mark Tried

MODIFY gt\_sud FROM gs\_sud INDEX sy-tabix TRANSPORTING pnp tnt.

\* suppress the options in other depending cells as per the tried option

PERFORM suppress\_in\_sud CHANGING chk.

ENDLOOP.

ENDFORM. " recursive\_fill

\*&---------------------------------------------------------------------\*

\*& Form FILL\_INITIAL\_POSSIBLE

\*&---------------------------------------------------------------------\*

\* text

\*----------------------------------------------------------------------\*

\* --> p1 text

\* <-- p2 text

\*----------------------------------------------------------------------\*

FORM fill\_initial\_possible .

DO 9 TIMES.

row = row + 1.

CLEAR: col, ava.

DO 9 TIMES.

col = col + 1.

CLEAR ava.

DO 9 TIMES.

ava = ava + 1.

gs\_sud-row = row.

gs\_sud-col = col.

gs\_sud-ava = ava.

\* CONCATENATE gs\_sud-row gs\_sud-col INTO gs\_sud-con.

IF row LE 3 AND col LE 3.

gs\_sud-blk = 1.

gs\_sud-seq = 1.

ENDIF.

IF row LE 3 AND ( col GE 4 AND col LE 6 ).

gs\_sud-blk = 2.

gs\_sud-seq = 3.

ENDIF.

IF row LE 3 AND col GE 7.

gs\_sud-blk = 3.

gs\_sud-seq = 2.

ENDIF.

\* next 3 rows.

IF ( row GE 4 AND row LE 6 ) AND col LE 3.

gs\_sud-blk = 4.

gs\_sud-seq = 3.

ENDIF.

IF ( row GE 4 AND row LE 6 ) AND ( col GE 4 AND col LE 6 ).

gs\_sud-blk = 5.

gs\_sud-seq = 1.

ENDIF.

IF ( row GE 4 AND row LE 6 ) AND col GE 7.

gs\_sud-blk = 6.

gs\_sud-seq = 3.

ENDIF.

\* last 3 rows.

IF row GE 7 AND col LE 3.

gs\_sud-blk = 7.

gs\_sud-seq = 2.

ENDIF.

IF row GE 7 AND ( col GE 4 AND col LE 6 ).

gs\_sud-blk = 8.

gs\_sud-seq = 3.

ENDIF.

IF row GE 7 AND col GE 7.

gs\_sud-blk = 9.

gs\_sud-seq = 1.

ENDIF.

CONCATENATE row col INTO gs\_sud-con.

APPEND gs\_sud TO gt\_sud.

ENDDO.

ENDDO.

ENDDO.

ENDFORM. " FILL\_INITIAL\_POSSIBLE

\*&---------------------------------------------------------------------\*

\*& Form FILL\_DIAGONAL\_BLOCKS

\*&---------------------------------------------------------------------\*

\* text

\*----------------------------------------------------------------------\*

\* --> p1 text

\* <-- p2 text

\*----------------------------------------------------------------------\*

FORM fill\_diagonal\_blocks .

\* All diagonal blocks are preloaded with seq value '1'

LOOP AT gt\_sud INTO gs\_sud WHERE seq = 1

AND val = '' .

CLEAR : row, col, ava.

row = gs\_sud-row.

col = gs\_sud-col.

FREE: gt\_sud1[].

gt\_sud1[] = gt\_sud[].

DELETE gt\_sud1 WHERE row NE row

OR col NE col.

DESCRIBE TABLE gt\_sud1 LINES lines.

lines = lines - 1.

\* getnerate a random number. " this FM give from 0 to lines.

CALL FUNCTION 'GENERAL\_GET\_RANDOM\_INT'

EXPORTING

range = lines

IMPORTING

random = random.

random = random + 1.

CLEAR gs\_sud1.

READ TABLE gt\_sud1 INTO gs\_sud1 INDEX random.

\* random possible value selected from the remaining possible

\* values of a particular cell

rand = gs\_sud1-ava.

CLEAR gs\_sud.

READ TABLE gt\_sud INTO gs\_sud WITH KEY row = row

col = col

ava = rand.

IF sy-subrc IS INITIAL.

index = sy-tabix.

gs\_sud-val = rand.

gs\_sud-ava = ''.

gs\_sud-tnt = 'T'.

gs\_sud-pnp = 'P'.

MODIFY gt\_sud FROM gs\_sud INDEX index TRANSPORTING val ava.

\* Keep the picked value and delete the rest of possible values

DELETE gt\_sud WHERE row = row

AND col = col

AND val = ''.

\* delete the picked value in the corresponding row from the

\* possible values in all cells belonging to that row.

DELETE gt\_sud WHERE row = row

AND ava = rand.

\* delete the picked value in the corresponding column from the

\* possible values in all cells belonging to that column.

DELETE gt\_sud WHERE col = col

AND ava = rand.

\* first 3 rows

\* delete the picked value in the corresponding block(3 x 3) from

\* the possible values in all cells belonging to that block \*

IF row LE 3 AND col LE 3.

DELETE gt\_sud WHERE row LE 3

AND col LE 3

AND ava = rand.

CONTINUE.

ENDIF.

IF row LE 3 AND ( col GE 4 AND col LE 6 ).

DELETE gt\_sud WHERE row LE 3

AND ( col GE 4 AND col LE 6 )

AND ava = rand.

CONTINUE.

ENDIF.

IF row LE 3 AND col GE 7.

DELETE gt\_sud WHERE row LE 3

AND col GE 7

AND ava = rand.

CONTINUE.

ENDIF.

\* next 3 rows.

IF ( row GE 4 AND row LE 6 ) AND col LE 3.

DELETE gt\_sud WHERE ( row GE 4 AND row LE 6 )

AND col LE 3

AND ava = rand.

CONTINUE.

ENDIF.

IF ( row GE 4 AND row LE 6 ) AND ( col GE 4 AND col LE 6 ).

DELETE gt\_sud WHERE ( row GE 4 AND row LE 6 )

AND ( col GE 4 AND col LE 6 )

AND ava = rand.

CONTINUE.

ENDIF.

IF ( row GE 4 AND row LE 6 ) AND col GE 7.

DELETE gt\_sud WHERE ( row GE 4 AND row LE 6 )

AND col GE 7

AND ava = rand.

CONTINUE.

ENDIF.

\* last 3 rows.

IF row GE 7 AND col LE 3.

DELETE gt\_sud WHERE row GE 7

AND col LE 3

AND ava = rand.

CONTINUE.

ENDIF.

IF row GE 7 AND ( col GE 4 AND col LE 6 ).

DELETE gt\_sud WHERE row GE 7

AND ( col GE 4 AND col LE 6 )

AND ava = rand.

CONTINUE.

ENDIF.

IF row GE 7 AND col GE 7.

DELETE gt\_sud WHERE row GE 7

AND col GE 7

AND ava = rand.

CONTINUE.

ENDIF.

ENDIF.

ENDLOOP.

DESCRIBE TABLE gt\_sud LINES lines.

\* unproceed indicator is used at a later stage

unprocessed = lines - 27 + 1. " 27 for already filled diagonal blocks

unsolved = 54.

ENDFORM. " FILL\_DIAGONAL\_BLOCKS

\*&---------------------------------------------------------------------\*

\*& Form DISPLAY\_SUD

\*&---------------------------------------------------------------------\*

\* text

\*----------------------------------------------------------------------\*

\* --> p1 text

\* <-- p2 text

\*----------------------------------------------------------------------\*

FORM display\_sud .

\* IF chk = ''.

\* WRITE : 'Unique Sudoku with recursions : ', tcount.

\* ELSE.

\* WRITE : 'Non-Unique Sudoku with recursions : ', tcount.

\* ENDIF.

WRITE : 'Recursions performed: ', tcount.

WRITE /.

IF p\_upl NE 'X'.

LOOP AT gt\_sud INTO gs\_sud.

IF sy-tabix = 1.

WRITE: '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'.

ENDIF.

IF gs\_sud-row = 4 AND gs\_sud-col = 1.

WRITE: /'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'.

\* WRITE /.

ENDIF.

IF gs\_sud-row = 7 AND gs\_sud-col = 1.

WRITE: / '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'.

\* WRITE /.

ENDIF.

ON CHANGE OF gs\_sud-row.

WRITE /. WRITE '|'.

CLEAR index.

ENDON.

gap = gs\_sud-col - index.

IF gap GT 1.

gap = gap - 1.

DO gap TIMES.

WRITE '-' .

ENDDO.

ENDIF.

index = gs\_sud-col.

WRITE gs\_sud-val.

IF gs\_sud-col = 3 OR gs\_sud-col = 6 OR gs\_sud-col = 9.

WRITE '|'.

ENDIF.

AT LAST.

WRITE: /'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'.

ENDAT.

CLEAR gs\_sud.

ENDLOOP.

ELSE.

WRITE: / 'Uploaded Sudoku:'.

WRITE:/.

LOOP AT gt\_sudiu INTO gs\_sud.

IF sy-tabix = 1.

WRITE: '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'.

ENDIF.

IF gs\_sud-row = 4 AND gs\_sud-col = 1.

WRITE: /'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'.

\* WRITE /.

ENDIF.

IF gs\_sud-row = 7 AND gs\_sud-col = 1.

WRITE: / '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'.

\* WRITE /.

ENDIF.

ON CHANGE OF gs\_sud-row.

WRITE /. WRITE '|'.

CLEAR index.

ENDON.

gap = gs\_sud-col - index.

IF gap GT 1.

gap = gap - 1.

DO gap TIMES.

WRITE '-' .

ENDDO.

ENDIF.

index = gs\_sud-col.

WRITE gs\_sud-val.

IF gs\_sud-col = 3 OR gs\_sud-col = 6 OR gs\_sud-col = 9.

WRITE '|'.

ENDIF.

AT LAST.

WRITE: /'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'.

ENDAT.

CLEAR gs\_sud.

ENDLOOP.

IF unsolvable NE 'X'.

WRITE:/.

WRITE:/.

WRITE: / 'Solved Sudoku:'.

WRITE:/.

LOOP AT gt\_sudf INTO gs\_sud.

IF sy-tabix = 1.

WRITE: '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'.

ENDIF.

IF gs\_sud-row = 4 AND gs\_sud-col = 1.

WRITE: /'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'.

\* WRITE /.

ENDIF.

IF gs\_sud-row = 7 AND gs\_sud-col = 1.

WRITE: / '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'.

\* WRITE /.

ENDIF.

ON CHANGE OF gs\_sud-row.

WRITE /. WRITE '|'.

CLEAR index.

ENDON.

gap = gs\_sud-col - index.

IF gap GT 1.

gap = gap - 1.

DO gap TIMES.

WRITE '-' .

ENDDO.

ENDIF.

index = gs\_sud-col.

WRITE gs\_sud-val.

IF gs\_sud-col = 3 OR gs\_sud-col = 6 OR gs\_sud-col = 9.

WRITE '|'.

ENDIF.

AT LAST.

WRITE: /'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'.

ENDAT.

CLEAR gs\_sud.

ENDLOOP.

ELSE.

WRITE:/.

WRITE: / 'The uploaded Sudoku is UnSolvable'.

ENDIF.

ENDIF.

ENDFORM. " DISPLAY\_SUD

\*&---------------------------------------------------------------------\*

\*& Form suppress\_in\_sud

\*&---------------------------------------------------------------------\*

\* text

\*----------------------------------------------------------------------\*

\* <--P\_CHK text

\*----------------------------------------------------------------------\*

FORM suppress\_in\_sud CHANGING chk.

DATA: lv\_index TYPE sy-tabix.

CLEAR chk.

CLEAR : gs\_sud2.

LOOP AT gt\_sud INTO gs\_sud2 WHERE row = gs\_sud-row

AND col = gs\_sud-col

AND tnt = ''

AND pnp = ''.

lv\_index = sy-tabix.

gs\_sud2-pnp = 'S'. "Suppressed

MODIFY gt\_sud FROM gs\_sud2 INDEX lv\_index TRANSPORTING pnp.

\* chk = 'X'.

CLEAR gs\_sud2.

ENDLOOP.

CLEAR : gs\_sud2.

LOOP AT gt\_sud INTO gs\_sud2 WHERE row = gs\_sud-row

AND ava = gs\_sud-ava

AND col GT gs\_sud-col

AND tnt = ''

AND pnp = ''.

lv\_index = sy-tabix.

gs\_sud2-pnp = 'N'.

MODIFY gt\_sud FROM gs\_sud2 INDEX lv\_index TRANSPORTING pnp.

\* chk = 'X'.

CLEAR gs\_sud2.

ENDLOOP.

CLEAR : gs\_sud2.

LOOP AT gt\_sud INTO gs\_sud2 WHERE col = gs\_sud-col

AND ava = gs\_sud-ava

AND row GT gs\_sud-row

AND tnt = ''

AND pnp = ''.

lv\_index = sy-tabix.

gs\_sud2-pnp = 'N'.

MODIFY gt\_sud FROM gs\_sud2 INDEX lv\_index TRANSPORTING pnp.

\* chk = 'X'.

CLEAR gs\_sud2.

ENDLOOP.

CLEAR : gs\_sud2.

LOOP AT gt\_sud INTO gs\_sud2 WHERE blk = gs\_sud-blk

AND ava = gs\_sud-ava

AND seq GT gs\_sud-seq

AND tnt = ''

AND pnp = ''.

lv\_index = sy-tabix.

gs\_sud2-pnp = 'N'.

MODIFY gt\_sud FROM gs\_sud2 INDEX lv\_index TRANSPORTING pnp.

\* chk = 'X'.

CLEAR gs\_sud2.

ENDLOOP.

ENDFORM. " suppress\_in\_sud

\*&---------------------------------------------------------------------\*

\*& Form FILL\_NEW\_SEQUENCE

\*&---------------------------------------------------------------------\*

\* text

\*----------------------------------------------------------------------\*

\* --> p1 text

\* <-- p2 text

\*----------------------------------------------------------------------\*

FORM fill\_new\_sequence .

SORT gt\_sud BY val con.

\* Now all the unsolved cells needs to be processed, before that

\* mark those cells accordingly for distinguishing

CLEAR seq.

LOOP AT gt\_sud INTO gs\_sud.

IF sy-tabix EQ 1.

nsq = 1.

con = gs\_sud-con.

ENDIF.

IF gs\_sud-con NE con.

nsq = nsq + 1.

con = gs\_sud-con.

ENDIF.

seq = seq + 1.

gs\_sud-seq = seq.

gs\_sud-nsq = nsq.

\* T means Tried, 'P' means possible values

IF gs\_sud-val IS NOT INITIAL.

gs\_sud-ava = gs\_sud-val.

gs\_sud-tnt = 'T'.

gs\_sud-pnp = 'P'.

ENDIF.

MODIFY gt\_sud FROM gs\_sud INDEX sy-tabix

TRANSPORTING seq nsq ava tnt pnp.

CLEAR gs\_sud.

ENDLOOP.

ENDFORM. " FILL\_NEW\_SEQUENCE

\*&---------------------------------------------------------------------\*

\*& Form UNCHECK\_IN\_SUD

\*&---------------------------------------------------------------------\*

\* text

\*----------------------------------------------------------------------\*

\* <--P\_CHK text

\*----------------------------------------------------------------------\*

FORM uncheck\_in\_sud CHANGING chk.

DATA: lv\_index TYPE sy-tabix.

LOOP AT gt\_sud INTO gs\_sud2 FROM sy-tabix WHERE tnt = 'T' AND val = ''.

IF gs\_sud2-val IS NOT INITIAL.

EXIT.

ENDIF.

IF gs\_sud2-nsq = nsq AND gs\_sud2-pnp = 'P'.

gs\_sud2-pnp = 'N'.

ELSE.

gs\_sud2-tnt = ''.

gs\_sud2-pnp = ''.

ENDIF.

MODIFY gt\_sud FROM gs\_sud2 INDEX sy-tabix TRANSPORTING tnt pnp.

CLEAR : gs\_sud1.

LOOP AT gt\_sud INTO gs\_sud1 WHERE row = gs\_sud2-row

AND ava = gs\_sud2-ava

AND col GT gs\_sud2-col

AND pnp = 'N'.

lv\_index = sy-tabix.

gs\_sud1-pnp = ''.

MODIFY gt\_sud FROM gs\_sud1 INDEX lv\_index TRANSPORTING pnp.

\* chk = 'X'.

CLEAR gs\_sud1.

ENDLOOP.

CLEAR : gs\_sud1.

LOOP AT gt\_sud INTO gs\_sud1 WHERE col = gs\_sud2-col

AND ava = gs\_sud2-ava

AND row GT gs\_sud2-row

AND pnp = 'N'.

lv\_index = sy-tabix.

gs\_sud1-pnp = ''.

MODIFY gt\_sud FROM gs\_sud1 INDEX lv\_index TRANSPORTING pnp.

\* chk = 'X'.

CLEAR gs\_sud1.

ENDLOOP.

CLEAR : gs\_sud1.

LOOP AT gt\_sud INTO gs\_sud1 WHERE blk = gs\_sud2-blk

AND ava = gs\_sud2-ava

AND seq GT gs\_sud2-seq

AND pnp = 'N'.

lv\_index = sy-tabix.

gs\_sud1-pnp = ''.

MODIFY gt\_sud FROM gs\_sud1 INDEX lv\_index TRANSPORTING pnp.

\* chk = 'X'.

CLEAR gs\_sud1.

ENDLOOP.

CLEAR gs\_sud2.

ENDLOOP.

ENDFORM. " UNCHECK\_IN\_SUD

\*&---------------------------------------------------------------------\*

\*& Form MAP\_COMPLETE\_POSSIBILITY

\*&---------------------------------------------------------------------\*

\* text

\*----------------------------------------------------------------------\*

\* --> p1 text

\* <-- p2 text

\*----------------------------------------------------------------------\*

FORM map\_complete\_possibility CHANGING chk.

DATA: lv\_lines TYPE sy-tabix,

lv\_index TYPE sy-tabix.

CLEAR chk.

gt\_sudf[] = gt\_sud[].

\* delete all the cells which are not Tried and not possible option

DELETE gt\_sudf WHERE tnt NE 'T' OR pnp NE 'P'.

DESCRIBE TABLE gt\_sudf LINES lv\_lines.

IF lv\_lines EQ 81.

FREE: gt\_sud[].

LOOP AT gt\_sudf INTO gs\_sud WHERE val EQ '' .

\* Finally fill the final derived value for that particular cell

gs\_sud-val = gs\_sud-ava.

CLEAR gs\_sud-ava.

MODIFY gt\_sudf FROM gs\_sud INDEX sy-tabix TRANSPORTING val ava.

CLEAR gs\_sud.

ENDLOOP.

SORT gt\_sudf BY row col.

ELSE.

FREE: gt\_sudf[].

\* cannot derive unique sudoku

chk = 'X'.

READ TABLE gt\_sud INTO gs\_sud2 WITH KEY val = '' tnt = 'T' pnp = 'P' .

IF sy-subrc IS INITIAL.

\* changed the possible option to Not possible and clear all

\* the marked categories accordingly and continue from there

gs\_sud2-pnp = 'N'.

MODIFY gt\_sud FROM gs\_sud2 INDEX sy-tabix TRANSPORTING pnp.

lv\_index = sy-tabix + 1.

LOOP AT gt\_sud INTO gs\_sud2 FROM lv\_index WHERE val = ''.

gs\_sud2-tnt = ''.

gs\_sud2-pnp = ''.

MODIFY gt\_sud FROM gs\_sud2 INDEX sy-tabix TRANSPORTING tnt pnp.

CLEAR gs\_sud2.

ENDLOOP.

ENDIF.

restart = 0.

ENDIF.

ENDFORM. " MAP\_COMPLETE\_POSSIBILITY

\*&---------------------------------------------------------------------\*

\*& Form CHECK\_IN\_SUD

\*&---------------------------------------------------------------------\*

\* text

\*----------------------------------------------------------------------\*

\* <--P\_CHK text

\*----------------------------------------------------------------------\*

FORM check\_in\_sud CHANGING chk.

DATA: lv\_index TYPE sy-tabix.

lv\_index = sy-tabix.

CLEAR : chk, gs\_sud2.

LOOP AT gt\_sud INTO gs\_sud2 WHERE row = gs\_sud-row

AND ava = gs\_sud-ava

AND col LT gs\_sud-col

AND tnt = 'T'

AND pnp = 'P'.

gs\_sud2-tnt = 'T'.

gs\_sud2-pnp = 'N'.

MODIFY gt\_sud FROM gs\_sud2 INDEX lv\_index TRANSPORTING tnt pnp.

chk = 'X'.

EXIT.

ENDLOOP.

IF chk = 'X'.

EXIT.

ENDIF.

CLEAR : chk, gs\_sud2.

LOOP AT gt\_sud INTO gs\_sud2 WHERE col = gs\_sud-col

AND ava = gs\_sud-ava

AND row LT gs\_sud-row

AND tnt = 'T'

AND pnp = 'P'.

gs\_sud2-tnt = 'T'.

gs\_sud2-pnp = 'N'.

MODIFY gt\_sud FROM gs\_sud2 INDEX lv\_index TRANSPORTING tnt pnp.

chk = 'X'.

EXIT.

ENDLOOP.

IF chk = 'X'.

EXIT.

ENDIF.

CLEAR : chk, gs\_sud2.

LOOP AT gt\_sud INTO gs\_sud2 WHERE blk = gs\_sud-blk

AND ava = gs\_sud-ava

AND seq LT gs\_sud-seq

AND tnt = 'T'

AND pnp = 'P'.

gs\_sud2-tnt = 'T'.

gs\_sud2-pnp = 'N'.

MODIFY gt\_sud FROM gs\_sud2 INDEX lv\_index TRANSPORTING tnt pnp.

chk = 'X'.

EXIT.

ENDLOOP.

ENDFORM. " CHECK\_IN\_SUD

\*&---------------------------------------------------------------------\*

\*& Form FINAL\_CHECK

\*&---------------------------------------------------------------------\*

\* text

\*----------------------------------------------------------------------\*

\* --> p1 text

\* <-- p2 text

\*----------------------------------------------------------------------\*

FORM final\_check CHANGING chk.

DATA: lv\_index TYPE sy-tabix.

lv\_index = sy-tabix.

CLEAR : chk, gs\_sud, gs\_sud2.

LOOP AT gt\_sud INTO gs\_sud.

LOOP AT gt\_sud INTO gs\_sud2 WHERE row = gs\_sud-row

AND val = gs\_sud-val

AND col NE gs\_sud-col.

chk = 'X'.

CLEAR gs\_sud2.

EXIT.

ENDLOOP.

IF chk = 'X'.

EXIT.

ENDIF.

LOOP AT gt\_sud INTO gs\_sud2 WHERE row NE gs\_sud-row

AND val = gs\_sud-val

AND col EQ gs\_sud-col.

chk = 'X'.

CLEAR gs\_sud2.

EXIT.

ENDLOOP.

IF chk = 'X'.

EXIT.

ENDIF.

LOOP AT gt\_sud INTO gs\_sud2 WHERE row NE gs\_sud-row

AND val = gs\_sud-val

AND col NE gs\_sud-col

AND blk = gs\_sud-blk.

chk = 'X'.

CLEAR gs\_sud2.

EXIT.

ENDLOOP.

IF chk = 'X'.

EXIT.

ENDIF.

ENDLOOP.

ENDFORM. " FINAL\_CHECK

\*&---------------------------------------------------------------------\*

\*& Form GET\_FILENAME

\*&---------------------------------------------------------------------\*

\* text

\*----------------------------------------------------------------------\*

\* --> p1 text

\* <-- p2 text

\*----------------------------------------------------------------------\*

FORM get\_filename .

CALL FUNCTION 'F4\_FILENAME'

EXPORTING

program\_name = syst-cprog

dynpro\_number = syst-dynnr

field\_name = ' '

IMPORTING

file\_name = p\_file.

ENDFORM.

\*&---------------------------------------------------------------------\*

\*& Form UPLOAD\_EXCEL

\*&---------------------------------------------------------------------\*

\* text

\*----------------------------------------------------------------------\*

\* --> p1 text

\* <-- p2 text

\*----------------------------------------------------------------------\*

FORM upload\_excel .

\* DATA: lv\_begcol TYPE i,

\* lv\_begrow TYPE i,

\* lv\_endcol TYPE i,

\* lv\_endrow TYPE i.

\* lv\_begcol = p\_begcol.

\* lv\_begrow = p\_begrow.

\* lv\_endcol = p\_endcol.

\* lv\_endrow = p\_endrow.

\* Get data into internal table from Excel file sheets

CALL FUNCTION 'ALSM\_EXCEL\_TO\_INTERNAL\_TABLE'

EXPORTING

filename = p\_file

i\_begin\_col = 1 "lv\_begcol

i\_begin\_row = 1 "lv\_begrow

i\_end\_col = 9 "lv\_endcol

i\_end\_row = 9 "lv\_endrow

TABLES

intern = itab\_upload\_file

EXCEPTIONS

inconsistent\_parameters = 1

upload\_ole = 2

OTHERS = 3.

IF sy-subrc <> 0.

MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno

WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.

ENDIF.

ENDFORM.

\*&---------------------------------------------------------------------\*

\*& Form CLEAR\_FEW\_VALUES

\*&---------------------------------------------------------------------\*

\* text

\*----------------------------------------------------------------------\*

\* --> p1 text

\* <-- p2 text

\*----------------------------------------------------------------------\*

FORM clear\_few\_values .

DO 75 TIMES.

\* get random number. " this FM give from 0 to lines.

CALL FUNCTION 'GENERAL\_GET\_RANDOM\_INT'

EXPORTING

range = 80 "lines

IMPORTING

random = random.

random = random + 1.

CLEAR gs\_sud.

READ TABLE gt\_sud INTO gs\_sud INDEX random.

IF sy-subrc IS INITIAL.

CLEAR gs\_sud-val.

MODIFY gt\_sud FROM gs\_sud INDEX random TRANSPORTING val.

ENDIF.

ENDDO.

ENDFORM.

\*&---------------------------------------------------------------------\*

\*& Form write\_sud\_FILE

\*&---------------------------------------------------------------------\*

\* text

\*----------------------------------------------------------------------\*

\* --> p1 text

\* <-- p2 text

\*----------------------------------------------------------------------\*

FORM write\_sud\_file .

LOOP AT gt\_sud INTO gs\_sud.

CASE gs\_sud-row.

WHEN '1'.

CASE gs\_sud-col.

WHEN '1'.

gs\_file1-c1 = gs\_sud-val.

WHEN '2'.

gs\_file1-c2 = gs\_sud-val.

WHEN '3'.

gs\_file1-c3 = gs\_sud-val.

WHEN '4'.

gs\_file1-c4 = gs\_sud-val.

WHEN '5'.

gs\_file1-c5 = gs\_sud-val.

WHEN '6'.

gs\_file1-c6 = gs\_sud-val.

WHEN '7'.

gs\_file1-c7 = gs\_sud-val.

WHEN '8'.

gs\_file1-c8 = gs\_sud-val.

WHEN '9'.

gs\_file1-c9 = gs\_sud-val.

ENDCASE.

WHEN '2'.

CASE gs\_sud-col.

WHEN '1'.

gs\_file2-c1 = gs\_sud-val.

WHEN '2'.

gs\_file2-c2 = gs\_sud-val.

WHEN '3'.

gs\_file2-c3 = gs\_sud-val.

WHEN '4'.

gs\_file2-c4 = gs\_sud-val.

WHEN '5'.

gs\_file2-c5 = gs\_sud-val.

WHEN '6'.

gs\_file2-c6 = gs\_sud-val.

WHEN '7'.

gs\_file2-c7 = gs\_sud-val.

WHEN '8'.

gs\_file2-c8 = gs\_sud-val.

WHEN '9'.

gs\_file2-c9 = gs\_sud-val.

ENDCASE.

WHEN '3'.

CASE gs\_sud-col.

WHEN '1'.

gs\_file3-c1 = gs\_sud-val.

WHEN '2'.

gs\_file3-c2 = gs\_sud-val.

WHEN '3'.

gs\_file3-c3 = gs\_sud-val.

WHEN '4'.

gs\_file3-c4 = gs\_sud-val.

WHEN '5'.

gs\_file3-c5 = gs\_sud-val.

WHEN '6'.

gs\_file3-c6 = gs\_sud-val.

WHEN '7'.

gs\_file3-c7 = gs\_sud-val.

WHEN '8'.

gs\_file3-c8 = gs\_sud-val.

WHEN '9'.

gs\_file3-c9 = gs\_sud-val.

ENDCASE.

WHEN '4'.

CASE gs\_sud-col.

WHEN '1'.

gs\_file4-c1 = gs\_sud-val.

WHEN '2'.

gs\_file4-c2 = gs\_sud-val.

WHEN '3'.

gs\_file4-c3 = gs\_sud-val.

WHEN '4'.

gs\_file4-c4 = gs\_sud-val.

WHEN '5'.

gs\_file4-c5 = gs\_sud-val.

WHEN '6'.

gs\_file4-c6 = gs\_sud-val.

WHEN '7'.

gs\_file4-c7 = gs\_sud-val.

WHEN '8'.

gs\_file4-c8 = gs\_sud-val.

WHEN '9'.

gs\_file4-c9 = gs\_sud-val.

ENDCASE.

WHEN '5'.

CASE gs\_sud-col.

WHEN '1'.

gs\_file5-c1 = gs\_sud-val.

WHEN '2'.

gs\_file5-c2 = gs\_sud-val.

WHEN '3'.

gs\_file5-c3 = gs\_sud-val.

WHEN '4'.

gs\_file5-c4 = gs\_sud-val.

WHEN '5'.

gs\_file5-c5 = gs\_sud-val.

WHEN '6'.

gs\_file5-c6 = gs\_sud-val.

WHEN '7'.

gs\_file5-c7 = gs\_sud-val.

WHEN '8'.

gs\_file5-c8 = gs\_sud-val.

WHEN '9'.

gs\_file5-c9 = gs\_sud-val.

ENDCASE.

WHEN '6'.

CASE gs\_sud-col.

WHEN '1'.

gs\_file6-c1 = gs\_sud-val.

WHEN '2'.

gs\_file6-c2 = gs\_sud-val.

WHEN '3'.

gs\_file6-c3 = gs\_sud-val.

WHEN '4'.

gs\_file6-c4 = gs\_sud-val.

WHEN '5'.

gs\_file6-c5 = gs\_sud-val.

WHEN '6'.

gs\_file6-c6 = gs\_sud-val.

WHEN '7'.

gs\_file6-c7 = gs\_sud-val.

WHEN '8'.

gs\_file6-c8 = gs\_sud-val.

WHEN '9'.

gs\_file6-c9 = gs\_sud-val.

ENDCASE.

WHEN '7'.

CASE gs\_sud-col.

WHEN '1'.

gs\_file7-c1 = gs\_sud-val.

WHEN '2'.

gs\_file7-c2 = gs\_sud-val.

WHEN '3'.

gs\_file7-c3 = gs\_sud-val.

WHEN '4'.

gs\_file7-c4 = gs\_sud-val.

WHEN '5'.

gs\_file7-c5 = gs\_sud-val.

WHEN '6'.

gs\_file7-c6 = gs\_sud-val.

WHEN '7'.

gs\_file7-c7 = gs\_sud-val.

WHEN '8'.

gs\_file7-c8 = gs\_sud-val.

WHEN '9'.

gs\_file7-c9 = gs\_sud-val.

ENDCASE.

WHEN '8'.

CASE gs\_sud-col.

WHEN '1'.

gs\_file8-c1 = gs\_sud-val.

WHEN '2'.

gs\_file8-c2 = gs\_sud-val.

WHEN '3'.

gs\_file8-c3 = gs\_sud-val.

WHEN '4'.

gs\_file8-c4 = gs\_sud-val.

WHEN '5'.

gs\_file8-c5 = gs\_sud-val.

WHEN '6'.

gs\_file8-c6 = gs\_sud-val.

WHEN '7'.

gs\_file8-c7 = gs\_sud-val.

WHEN '8'.

gs\_file8-c8 = gs\_sud-val.

WHEN '9'.

gs\_file8-c9 = gs\_sud-val.

ENDCASE.

WHEN '9'.

CASE gs\_sud-col.

WHEN '1'.

gs\_file9-c1 = gs\_sud-val.

WHEN '2'.

gs\_file9-c2 = gs\_sud-val.

WHEN '3'.

gs\_file9-c3 = gs\_sud-val.

WHEN '4'.

gs\_file9-c4 = gs\_sud-val.

WHEN '5'.

gs\_file9-c5 = gs\_sud-val.

WHEN '6'.

gs\_file9-c6 = gs\_sud-val.

WHEN '7'.

gs\_file9-c7 = gs\_sud-val.

WHEN '8'.

gs\_file9-c8 = gs\_sud-val.

WHEN '9'.

gs\_file9-c9 = gs\_sud-val.

ENDCASE.

ENDCASE.

ENDLOOP.

gs\_file1-r = 'row1'.

APPEND gs\_file1 TO gt\_file.

gs\_file2-r = 'row2'.

APPEND gs\_file2 TO gt\_file.

gs\_file3-r = 'row3'.

APPEND gs\_file3 TO gt\_file.

gs\_file4-r = 'row4'.

APPEND gs\_file4 TO gt\_file.

gs\_file5-r = 'row5'.

APPEND gs\_file5 TO gt\_file.

gs\_file6-r = 'row6'.

APPEND gs\_file6 TO gt\_file.

gs\_file7-r = 'row7'.

APPEND gs\_file7 TO gt\_file.

gs\_file8-r = 'row8'.

APPEND gs\_file8 TO gt\_file.

gs\_file9-r = 'row9'.

APPEND gs\_file9 TO gt\_file.

gs\_filec-c1 = 'col1'.

gs\_filec-c2 = 'col2'.

gs\_filec-c3 = 'col3'.

gs\_filec-c4 = 'col4'.

gs\_filec-c5 = 'col5'.

gs\_filec-c6 = 'col6'.

gs\_filec-c7 = 'col7'.

gs\_filec-c8 = 'col8'.

gs\_filec-c9 = 'col9'.

APPEND gs\_filec TO gt\_file.

DATA: lv\_filename TYPE string.

lv\_filename = p\_file.

SPLIT lv\_filename AT '.' INTO DATA(str1) DATA(str2).

CONCATENATE lv\_filename '.xls' INTO lv\_filename.

CALL METHOD cl\_gui\_frontend\_services=>gui\_download

EXPORTING

filename = lv\_filename

write\_field\_separator = 'X'

CHANGING

data\_tab = gt\_file.

ENDFORM.

\*&---------------------------------------------------------------------\*

\*& Form FILL\_FROM\_UPLOAD

\*&---------------------------------------------------------------------\*

\* text

\*----------------------------------------------------------------------\*

\* --> p1 text

\* <-- p2 text

\*----------------------------------------------------------------------\*

FORM fill\_from\_upload .

DESCRIBE TABLE itab\_upload\_file LINES lines.

given = lines.

" generally less than 17 values in an initial sudoku is unsolvable

IF given LT 17.

unsolvable = 'X'.

ENDIF.

unsolved = 81 - lines. " unsolved number of cells

SORT itab\_upload\_file BY row col value.

\* loop at uploaded file

LOOP AT itab\_upload\_file.

CLEAR : row, col, ava.

SHIFT itab\_upload\_file-row LEFT DELETING LEADING '0'.

row = itab\_upload\_file-row.

SHIFT itab\_upload\_file-col LEFT DELETING LEADING '0'.

col = itab\_upload\_file-col.

rand = itab\_upload\_file-value.

MODIFY itab\_upload\_file.

CLEAR gs\_sud2.

gs\_sud2-row = row.

gs\_sud2-col = col.

gs\_sud2-val = rand.

CONCATENATE row col INTO gs\_sud2-con.

"transfer the contents of uploaded file to an initial upload internal table

APPEND gs\_sud2 TO gt\_sudiu.

\* as per the initial value of a particular cell,

\*delete the same from other columns for that particular row

\*delete the same from other rows for that particular column

\*delete the same from the 3 x 3 block if any duplicate

CLEAR gs\_sud.

READ TABLE gt\_sud INTO gs\_sud WITH KEY row = row

col = col

ava = rand.

IF sy-subrc IS INITIAL.

index = sy-tabix.

gs\_sud-val = rand.

gs\_sud-ava = ''.

gs\_sud-tnt = 'T'.

gs\_sud-pnp = 'P'.

MODIFY gt\_sud FROM gs\_sud INDEX index TRANSPORTING val ava.

DELETE gt\_sud WHERE row = row

AND col = col

AND val = ''.

DELETE gt\_sud WHERE row = row

AND ava = rand.

DELETE gt\_sud WHERE col = col

AND ava = rand.

\* first 3 rows

IF row LE 3 AND col LE 3.

DELETE gt\_sud WHERE row LE 3

AND col LE 3

AND ava = rand.

CONTINUE.

ENDIF.

IF row LE 3 AND ( col GE 4 AND col LE 6 ).

DELETE gt\_sud WHERE row LE 3

AND ( col GE 4 AND col LE 6 )

AND ava = rand.

CONTINUE.

ENDIF.

IF row LE 3 AND col GE 7.

DELETE gt\_sud WHERE row LE 3

AND col GE 7

AND ava = rand.

CONTINUE.

ENDIF.

\* next 3 rows.

IF ( row GE 4 AND row LE 6 ) AND col LE 3.

DELETE gt\_sud WHERE ( row GE 4 AND row LE 6 )

AND col LE 3

AND ava = rand.

CONTINUE.

ENDIF.

IF ( row GE 4 AND row LE 6 ) AND ( col GE 4 AND col LE 6 ).

DELETE gt\_sud WHERE ( row GE 4 AND row LE 6 )

AND ( col GE 4 AND col LE 6 )

AND ava = rand.

CONTINUE.

ENDIF.

IF ( row GE 4 AND row LE 6 ) AND col GE 7.

DELETE gt\_sud WHERE ( row GE 4 AND row LE 6 )

AND col GE 7

AND ava = rand.

CONTINUE.

ENDIF.

\* last 3 rows.

IF row GE 7 AND col LE 3.

DELETE gt\_sud WHERE row GE 7

AND col LE 3

AND ava = rand.

CONTINUE.

ENDIF.

IF row GE 7 AND ( col GE 4 AND col LE 6 ).

DELETE gt\_sud WHERE row GE 7

AND ( col GE 4 AND col LE 6 )

AND ava = rand.

CONTINUE.

ENDIF.

IF row GE 7 AND col GE 7.

DELETE gt\_sud WHERE row GE 7

AND col GE 7

AND ava = rand.

CONTINUE.

ENDIF.

ENDIF.

CLEAR itab\_upload\_file.

ENDLOOP.

\* Now gt\_sud contains all possible values for a

\* particular cell if the cell is not filled from the upload

DESCRIBE TABLE gt\_sud LINES lines.

unprocessed = lines - given + 1. " 27 for already filled diagonal blocks

\* fill the rest cells with space such that

\*the sudoku can be printed as uploaded sudoku

CLEAR: row, col.

DO 9 TIMES.

row = row + 1.

CLEAR col.

DO 9 TIMES.

col = col + 1.

READ TABLE gt\_sudiu WITH KEY row = row

col = col TRANSPORTING NO FIELDS.

IF sy-subrc IS NOT INITIAL.

CLEAR gs\_sud.

gs\_sud-row = row.

gs\_sud-col = col.

CONCATENATE row col INTO gs\_sud-con.

APPEND gs\_sud TO gt\_sudiu.

ENDIF.

ENDDO.

ENDDO.

SORT gt\_sudiu BY row col val.

ENDFORM.

\*&---------------------------------------------------------------------\*

\*& Form INITIAL\_CHECK

\*&---------------------------------------------------------------------\*

\* text

\*----------------------------------------------------------------------\*

\* --> p1 text

\* <-- p2 text

\*----------------------------------------------------------------------\*

FORM initial\_check CHANGING chk.

DATA: lv\_index TYPE sy-tabix.

lv\_index = sy-tabix.

itab\_upload\_file\_temp[] = itab\_upload\_file[].

CLEAR : chk.

LOOP AT itab\_upload\_file.

LOOP AT itab\_upload\_file\_temp WHERE row = itab\_upload\_file-row

AND value = itab\_upload\_file-value

AND col NE itab\_upload\_file-col.

chk = 'X'.

CLEAR itab\_upload\_file\_temp.

EXIT.

ENDLOOP.

IF chk = 'X'.

EXIT.

ENDIF.

LOOP AT itab\_upload\_file\_temp WHERE row NE itab\_upload\_file-row

AND value = itab\_upload\_file-value

AND col EQ itab\_upload\_file-col.

chk = 'X'.

CLEAR itab\_upload\_file\_temp.

EXIT.

ENDLOOP.

IF chk = 'X'.

EXIT.

ENDIF.

LOOP AT itab\_upload\_file\_temp WHERE row NE itab\_upload\_file-row

AND value = itab\_upload\_file-value

AND col NE itab\_upload\_file-col.

IF itab\_upload\_file-row LE 3 AND itab\_upload\_file\_temp-row LE 3

AND itab\_upload\_file-col LE 3 AND itab\_upload\_file\_temp-col LE 3.

chk = 'X'.

CLEAR itab\_upload\_file\_temp.

EXIT.

ENDIF.

IF itab\_upload\_file-row LE 6 AND itab\_upload\_file-row GE 4

AND itab\_upload\_file-col LE 6 AND itab\_upload\_file-col GE 4

AND itab\_upload\_file\_temp-row LE 6 AND itab\_upload\_file\_temp-row GE 4

AND itab\_upload\_file\_temp-col LE 6 AND itab\_upload\_file\_temp-col GE 4.

chk = 'X'.

CLEAR itab\_upload\_file\_temp.

EXIT.

ENDIF.

IF itab\_upload\_file-row GE 7 AND itab\_upload\_file\_temp-row GE 7

AND itab\_upload\_file-col GE 7 AND itab\_upload\_file\_temp-col GE 7.

chk = 'X'.

CLEAR itab\_upload\_file\_temp.

EXIT.

ENDIF.

ENDLOOP.

IF chk = 'X'.

EXIT.

ENDIF.

ENDLOOP.

ENDFORM.