```
*&---------------------------------------------------------------------*
*& 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.
    PERFORM initial_check CHANGING chk. " check for duplicacy from the initial upload.
    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 fille
    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 playe
    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(
    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 per
  IF restart = 0 .
    restart = 1.
  ENDIF.

* unprocessed number is the already fixed value from the initial upload or randomly generated diago
* 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
    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 c
    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 particul
          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 particul
* 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
      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 th
* also unmark the not possible values from other cells and columns because we have concluded that t
        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 unspppressed 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
      DELETE gt_sud WHERE row = row
                    AND   ava = rand.
* delete the picked value in the corresponding column from the possible values in all cells belongi
      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 b
      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 di
  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 c
      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.

  IF given LT 17.   " generally less than 17 values in an initial sudoku is unsolvable
    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.
    APPEND gs_sud2 TO gt_sudiu.  "transfer the contents of uploaded file to an initial upload inter

* as per the initial value of a particular cell, delete the same from other columns for that partic
*                                                  delete the same from other rows for that particula
*                                                  delete the same from the 3 x 3 block if any duplic
    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
  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
        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
        chk = 'X'.
        CLEAR itab_upload_file_temp.
        EXIT.
      ENDIF.

    ENDLOOP.

    IF chk = 'X'.
      EXIT.
    ENDIF.

  ENDLOOP.


ENDFORM.
```