NAME – ISHIKA GOEL

UNIVERSITY ROLL NO – 19044563039

COLLEGE ROLL NO – MAT/19/93

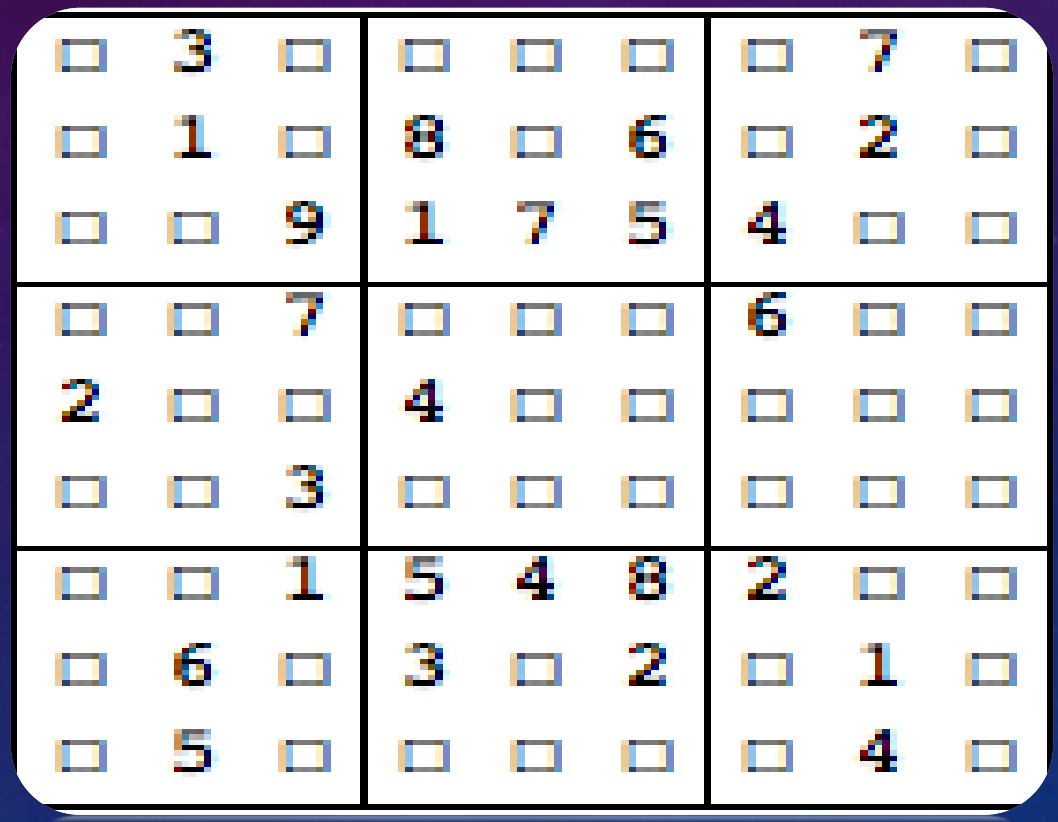
COURSE NAME –B.SC(HONS)MATHEMATICS

SUDOKU ON MATHEMATICA

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

Sudoku, for those unfamiliar with this puzzle, consists of a 9X9 square grid with nine 3X3 subgrids. The 81 entries are to be filled with the integers 1 to 9 in such a way that each row, column and subgrid contains all the nine integers. Some of the entries are already chosen, and the final puzzle solution must contain these initial choices.

We will work on the making and then solving the below Sudoku



FIRST STEPS TO SOLVING SUDOKU

The input for this puzzle is a list of nine lists consisting of blanks (shown as □) or integers between 1 and 9. A list of lists of the same length is regarded as a matrix in Mathematica, so we input for the puzzle and then show it in matrix form.

Input -

```
b = {{□, 3, □, □, □, □, □, 7, □}, {□, 1, □, 8, □, 6, □, 2, □}, {□, □, 9, 1, 7, 5, 4, □, □},
    {□, □, 7, □, □, □, 6, □, □}, {2, □, □, 4, □, 9, □, □, 3}, {□, □, 3, □, □, □, 1, □, □},
    {□, □, 1, 5, 4, 8, 2, □, □}, {□, 6, □, 3, □, 2, □, 1, □}, {□, 5, □, □, □, □, □, 4, □}};
MatrixForm[b]
```

Output -

$$\begin{pmatrix} □ & 3 & □ & □ & □ & □ & □ & 7 & □ \\ □ & 1 & □ & 8 & □ & 6 & □ & 2 & □ \\ □ & □ & 9 & 1 & 7 & 5 & 4 & □ & □ \\ □ & □ & 7 & □ & □ & □ & 6 & □ & □ \\ 2 & □ & □ & 4 & □ & 9 & □ & □ & 3 \\ □ & □ & 3 & □ & □ & □ & 1 & □ & □ \\ □ & □ & 1 & 5 & 4 & 8 & 2 & □ & □ \\ □ & 6 & □ & 3 & □ & 2 & □ & 1 & □ \\ □ & 5 & □ & □ & □ & □ & □ & 4 & □ \end{pmatrix}$$

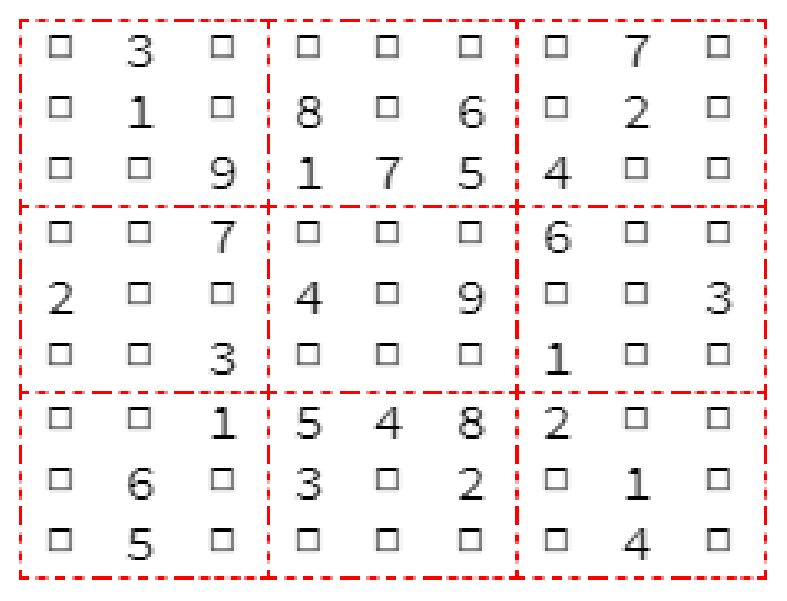
WE CAN ALSO DISPLAY THIS IN SUDOKU FORMAT BY DRAWING COLUMN AND ROW LINES AND A FRAME ■

Input -

```
display[X_] := Grid[Map[If[ListQ@#, Row[#], #] &, X, {2}],
 Frame → True, Dividers →
 {{True, False, False, True, False, False, True, False},
  {True, False, False, True, False, False, True, False}},
 FrameStyle → Directive[Red, Dotted]]

display[b]
```

Output -



Commands used-

- **Grid** - [{{$expr_{11}$,$expr_{12}$,…},{$expr_{21}$,$expr_{22}$,…},…}] is an object that formats with the $expr_{ij}$ arranged in a two-dimensional grid
- **Map** - represents an operator form of Map that can be applied to an expression.
- **ListQ[*expr*]** - gives True if the head of *expr* is List, and False otherwise.
- **Frame** -is an option for Graphics, Grid, and other constructs that specifies whether to include a frame.
- **Dividers** - is an option for Grid and related constructs that specifies where and how to draw divider lines.
- **FrameStyle** -is an option for Graphics, Grid, and other constructs that specifies the style in which to draw frames.
- **# and & -** The # symbol serves as the placeholder for the variable, while the & symbol precedes the value you wish to substitute into the function.
- **If** - If[*condition*,*t*,*f*] - gives *t* if *condition* evaluates to True, and *f* if it evaluates to False.

Now we will define a function *block[x_, i_ ,j_]* that gives a list of the entries that comprise the block of *X[[i, j]]* in X.

Input -

```
block[X_, i_, j_] := Which[
 1 ≤ i ≤ 3 && 1 ≤ j ≤ 3, Take[X, {1, 3}, {1, 3}],
 4 ≤ i ≤ 6 && 1 ≤ j ≤ 3, Take[X, {4, 6}, {1, 3}],
 7 ≤ i ≤ 9 && 1 ≤ j ≤ 3, Take[X, {7, 9}, {1, 3}],
 1 ≤ i ≤ 3 && 4 ≤ j ≤ 6, Take[X, {1, 3}, {4, 6}],
 4 ≤ i ≤ 6 && 4 ≤ j ≤ 6, Take[X, {4, 6}, {4, 6}],
 7 ≤ i ≤ 9 && 4 ≤ j ≤ 6, Take[X, {7, 9}, {4, 6}],
 1 ≤ i ≤ 3 && 7 ≤ j ≤ 9, Take[X, {1, 3}, {7, 9}],
 4 ≤ i ≤ 6 && 7 ≤ j ≤ 9, Take[X, {4, 6}, {7, 9}],
 7 ≤ i ≤ 9 && 7 ≤ j ≤ 9, Take[X, {7, 9}, {7, 9}]]

c = ReplaceAll[b, □ → Range[9]]


display[c]
```

Command Used –

- **Which**[$test_1$,$value_1$,$test_2$,$value_2$,…] - evaluates each of the $test_i$ in turn, returning the value of the $value_i$ corresponding to the first one that yields True.
- **Take**[***list*,*seq*$_1$,*seq*$_2$,…]** - gives a nested list in which elements specified by seq_i are taken at level *i* in *list*.
- **&&** - e_1&&e_2&&…is the logical AND function. It evaluates its arguments in order, giving False immediately if any of them are False, and True if they are all True
- **ReplaceAll** - applies a rule or list of rules in an attempt to transform each subpart of an expression *expr*.
- **Range** - Range[i_{max}]generates the list {1,2,…,i_{max}}.

Output -

Out[ ]= {{{1, 2, 3, 4, 5, 6, 7, 8, 9}, 3, {1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}, 7, {1, 2, 3, 4, 5, 6, 7, 8, 9}}, {{1, 2, 3, 4, 5, 6, 7, 8, 9}, 1, {1, 2, 3, 4, 5, 6, 7, 8, 9}, 8, {1, 2, 3, 4, 5, 6, 7, 8, 9}, 6, {1, 2, 3, 4, 5, 6, 7, 8, 9}, 2, {1, 2, 3, 4, 5, 6, 7, 8, 9}}, {{1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}, 9, 1, 7, 5, 4, {1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}}, {{1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}, 7, {1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}, 6, {1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}}, {2, {1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}, 4, {1, 2, 3, 4, 5, 6, 7, 8, 9}, 9, {1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}, 3}, {{1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}, 3, {1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}, 1, {1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}}, {{1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}, 1, 5, 4, 8, 2, {1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}}, {{1, 2, 3, 4, 5, 6, 7, 8, 9}, 6, {1, 2, 3, 4, 5, 6, 7, 8, 9}, 3, {1, 2, 3, 4, 5, 6, 7, 8, 9}, 2, {1, 2, 3, 4, 5, 6, 7, 8, 9}, 1, {1, 2, 3, 4, 5, 6, 7, 8, 9}}, {{1, 2, 3, 4, 5, 6, 7, 8, 9}, 5, {1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 4, 5, 6, 7, 8, 9}, 4, {1, 2, 3, 4, 5, 6, 7, 8, 9}}}

Out[ ]=

123456789	3	123456789	123456789	123456789	123456789	123456789	7	123456789
123456789	1	123456789	8	123456789	6	123456789	2	123456789
123456789	123456789	9	1	7	5	4	123456789	123456789
123456789	123456789	7	123456789	123456789	123456789	6	123456789	123456789
2	123456789	123456789	4	123456789	9	123456789	123456789	3
123456789	123456789	3	123456789	123456789	123456789	1	123456789	123456789
123456789	123456789	1	5	4	8	2	123456789	123456789
123456789	6	123456789	3	123456789	2	123456789	1	123456789
123456789	5	123456789	123456789	123456789	123456789	123456789	4	123456789

Our next task is to start eliminating candidate values in the entries that are lists of numbers in X, proceeding one entry `X[[i, j]]` at a time

```
DeleteSingletonsFromLists[X_] := Module[{A = X, integers},
  Table[
   integers = Select[
     Join[A[[i]], A[[All, j]], Flatten[block[A, i, j], 1]],
     IntegerQ[#] &
    ];
   If[ListQ[A[[i, j]]], A[[i, j]] = Complement[A[[i, j]], integers]];
   If[Length[A[[i, j]]] == 1, A[[i, j]] = First[A[[i, j]]]],
   {i, 9}, {j, 9}];
  A]
```

```
DeleteSingletonsFromLists[c] // display
```

4568	3	24568	29	29	4	589	7	15689
457	1	45	8	39	6	359	2	59
68	28	9	1	7	5	4	368	68
14589	489	7	2	1358	13	6	589	4589
2	8	56	4	156	9	57	5	3
4569	49	3	67	568	7	1	89	2489
379	79	1	5	4	8	2	369	679
4789	6	48	3	9	2	578	1	578
3789	5	28	67	16	1	3789	4	6789

Command Used -

- **Module**[{$x=x_0,\ldots$},*expr*]-defines initial values for *x*, ….
- **Join**[$list_1,list_2,\ldots,n$] -joins the objects at level *n* in each of the $list_i$
- **Flatten**[*list*,*n*]-flattens to level *n*.
- **ListQ**[*expr*]- gives True if the head of *expr* is List, and False otherwise
- **Complement**[$e_{all},e_1,e_2,\ldots$]-gives the elements in e_{all} that are not in any of the e_i.
- **Length**[*expr*]-gives the number of elements in *expr*
- **First**[*expr*,*def*]- gives the first element if it exists, or *def* otherwise
- **Table**[*expr*,*n*]- generates a list of *n* copies of *expr*.

To apply `DeleteSingletonsFromLists` again and again to `c` until the result no longer changes, we use `FixedPoint`

Input -

```
(d = FixedPoint[DeleteSingletonsFromLists, c]) // display
```

Output-

Out[ ]=

568	3	58	9	2	4	58	7	1568
457	1	45	8	3	6	59	2	59
68	2	9	1	7	5	4	368	68
1459	49	7	2	58	3	6	89	489
2	8	6	4	1	9	7	5	3
459	49	3	6	58	7	1	89	2489
379	79	1	5	4	8	2	369	679
478	6	48	3	9	2	58	1	578
389	5	28	7	6	1	389	4	89

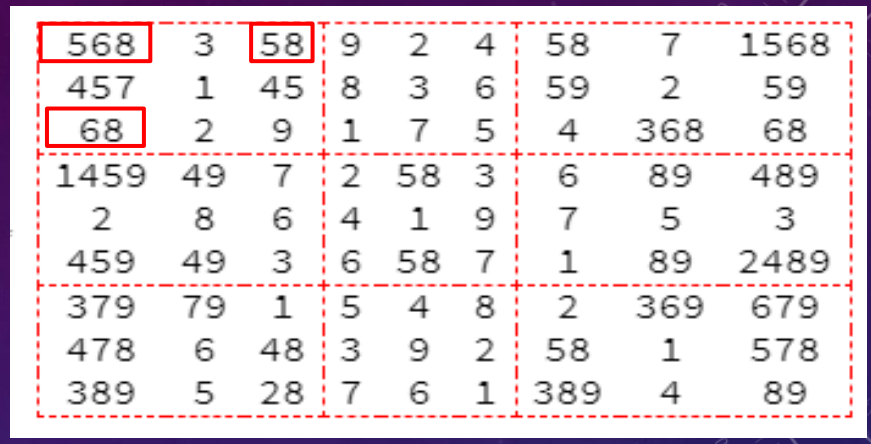
Command Used –

- **FixedPoint[*f*,*expr*]** - starts with *expr*, then applies *f* repeatedly until the result no longer changes.
- // - These notations extend to any function and any kind of argument:

but we see that we are still not done!

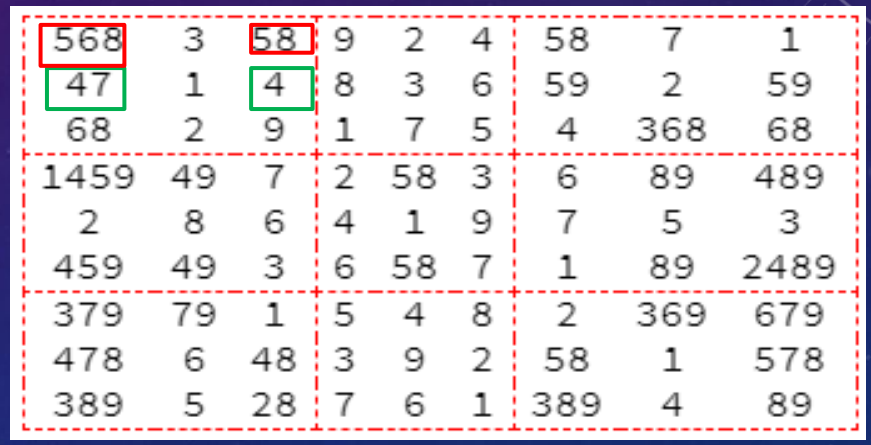
However, the first block has three entries (non –dotted red box) that are all sublists of {5,6,8}.

However, the first block has three entries (non - dotted red box) that are all sublists of



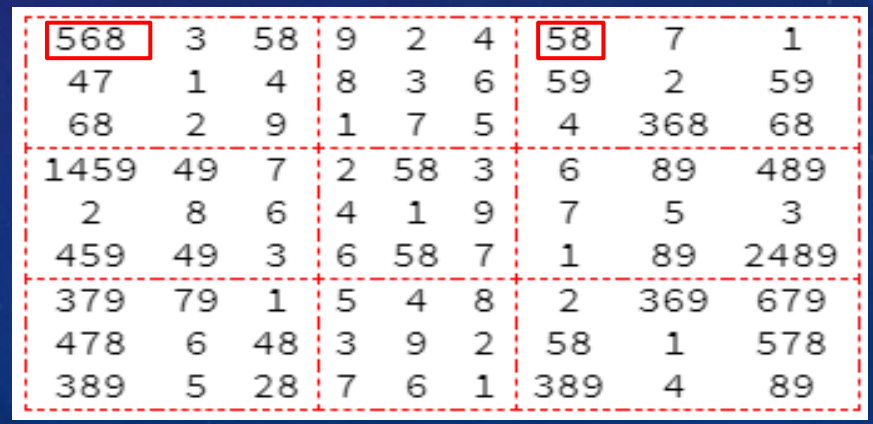
568	3	58	9	2	4	58	7	1568
457	1	45	8	3	6	59	2	59
68	2	9	1	7	5	4	368	68
1459	49	7	2	58	3	6	89	489
2	8	6	4	1	9	7	5	3
459	49	3	6	58	7	1	89	2489
379	79	1	5	4	8	2	369	679
478	6	48	3	9	2	58	1	578
389	5	28	7	6	1	389	4	89

While we do not know the exact value of any of the red entries, we know that the three numbers 5, 6 and 8 will be used up filling them; thus we can remove 5, 6 and 8 from the *other* entries in this block (non – dotted green box).



568	3	58	9	2	4	58	7	1
47	1	4	8	3	6	59	2	59
68	2	9	1	7	5	4	368	68
1459	49	7	2	58	3	6	89	489
2	8	6	4	1	9	7	5	3
459	49	3	6	58	7	1	89	2489
379	79	1	5	4	8	2	369	679
478	6	48	3	9	2	58	1	578
389	5	28	7	6	1	389	4	89

Similarly, in the first row, there are three entries that are sublists of {5,6,8}, so we remove 5, 6 and 8 from {1,5,6,8}, at the end of row 1; this defines e .



568	3	58	9	2	4	58	7	1
47	1	4	8	3	6	59	2	59
68	2	9	1	7	5	4	368	68
1459	49	7	2	58	3	6	89	489
2	8	6	4	1	9	7	5	3
459	49	3	6	58	7	1	89	2489
379	79	1	5	4	8	2	369	679
478	6	48	3	9	2	58	1	578
389	5	28	7	6	1	389	4	89

Thus we will perform this by -

Input -

```
(e = {{{5, 6, 8}, 3, {5, 8}, 9, 2, 4, {5, 8}, 7, 1}, {{4, 7}, 1, 4, 8, 3, 6, {5, 9}, 2, {5, 9}},
     {{6, 8}, 2, 9, 1, 7, 5, 4, {3, 6, 8}, {6, 8}},
     {{1, 4, 5, 9}, {4, 9}, 7, 2, {5, 8}, 3, 6, {8, 9}, {4, 8, 9}}, {2, 8, 6, 4, 1, 9, 7, 5, 3},
     {{4, 5, 9}, {4, 9}, 3, 6, {5, 8}, 7, 1, {8, 9}, {2, 4, 8, 9}},
     {{3, 7, 9}, {7, 9}, 1, 5, 4, 8, 2, {3, 6, 9}, {6, 7, 9}},
     {{4, 7, 8}, 6, {4, 8}, 3, 9, 2, {5, 8}, 1, {5, 7, 8}},
     {{3, 8, 9}, 5, {2, 8}, 7, 6, 1, {3, 8, 9}, 4, {8, 9}}}) // display
```

Output -

568	3	58	9	2	4	58	7	1
47	1	4	8	3	6	59	2	59
68	2	9	1	7	5	4	368	68
1459	49	7	2	58	3	6	89	489
2	8	6	4	1	9	7	5	3
459	49	3	6	58	7	1	89	2489
379	79	1	5	4	8	2	369	679
478	6	48	3	9	2	58	1	578
389	5	28	7	6	1	389	4	89

Then we use `FixedPoint` again and display the result.

Input -

```
FixedPoint[DeleteSingletonsFromLists, e] // display
```

Output -

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

We are done!

To Have a view on Mathematica -

https://www.wolframcloud.com/obj/847f2668-cbfe-4bda-a0dc-7289dc271a96

THANK YOU