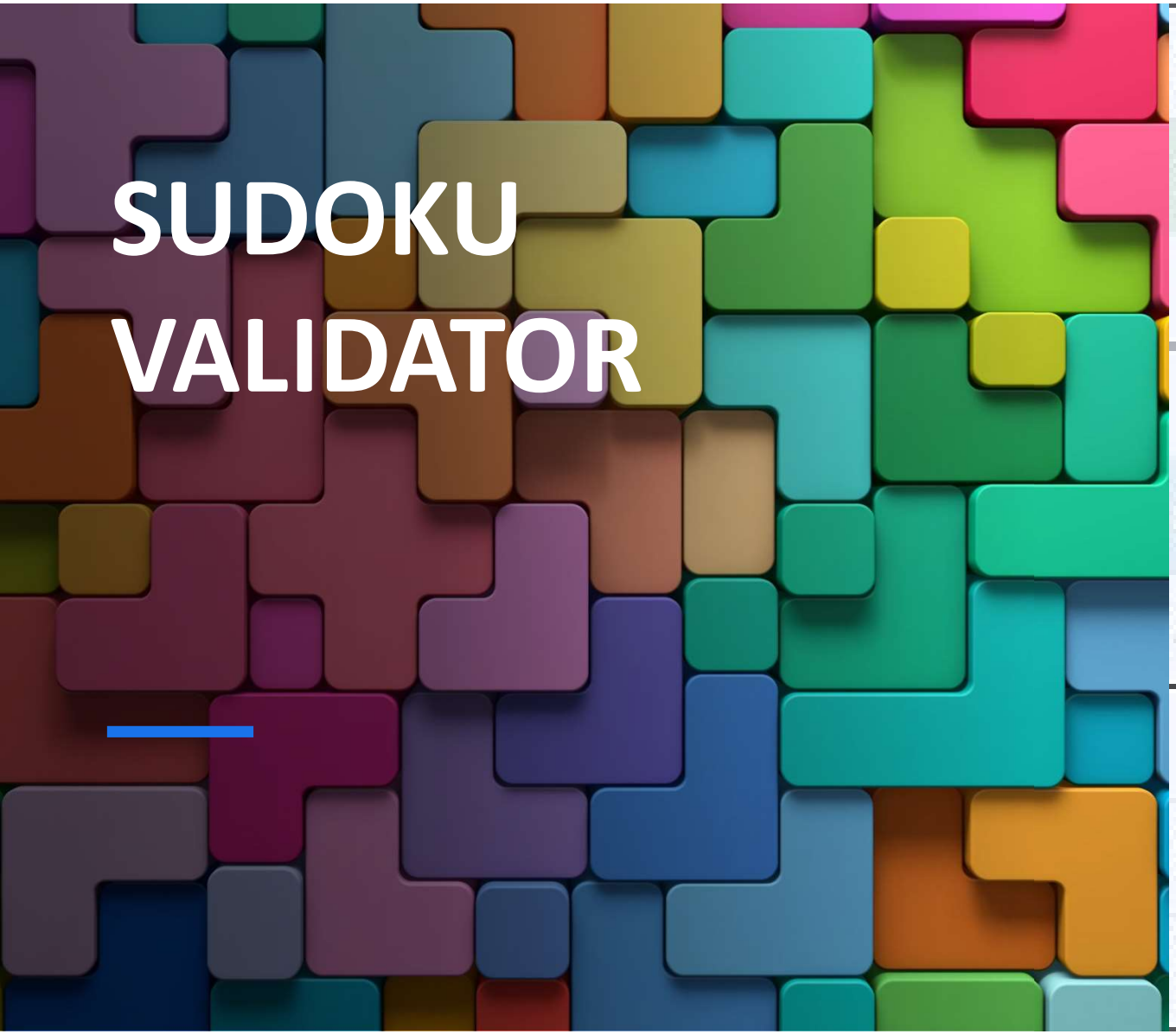
SUDOKU VALIDATOR

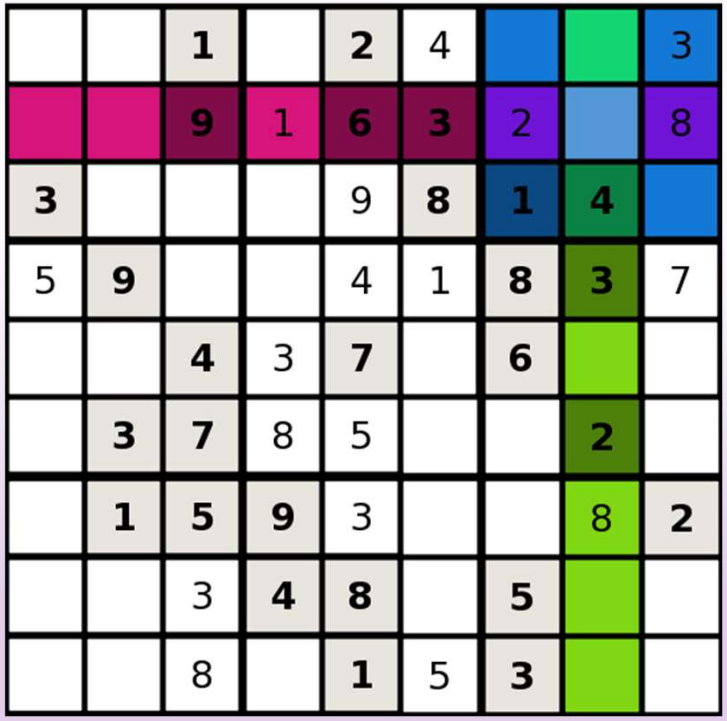


3	6	5		4	
5				1	
	4		8	2	
			3		
6			5	7	
4					
	5			3	
8		1		5	2
7		6			

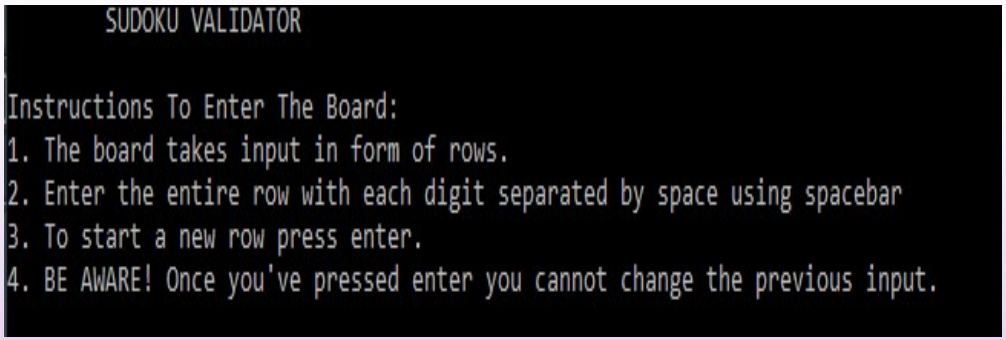
SUDOKU AND HOW IT IS PLAYED

- Sudoku is a very popular Japanese puzzle game. It is a game of logic; It is based on the logical placement of numbers.
- In this game, the player is given a 9x9 grid, divided into nine 3x3 sub-grids.
- The user is provided with a few boxes of some sub-grids of the board already filled with numbers from 1 to 9, and the user has to complete it while following sudoku rules, which are:

- Only digits from 1 to 9 can be used.
- There should be no repetition of any digit in any row.
- There should be no repetition of any digit in any column.
- There should be no repetition of any digit in any 3x3 sub-grid



HOW A SODOKU VALIDATOR HELPS



As difficulty level of the puzzle increases manual validation becomes:

- **More time consuming.**
- **More difficult.**
- **More prone to errors.**

Sudoku validator will help the player verify their board easily, within a few seconds and without human errors.

UNSOLVED:

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

SOLVED:

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

```c
#include <stdio.h>
int main (){
printf ("\tSUDOKU VALIDATOR\n\n");
printf ("Instructions To Enter The Board:\n1. The board takes input in form of rows.\n2. Enter the entire row with each d
int arr[9][9];
for (int r=0; r<9; r++){
for(int c=0; c<9; c++){
scanf ("%d", &arr[r][c]);
if (arr[r][c]<1 || arr[r][c]>9){
printf("INVALID INPUT!! SUDOKU RULE VIOLATED!! Number must be between 1-9");
return 0;
}
}
}
printf("\nRESU
// checking ro
for (int r=0; r<9; r++){
int check_r[9]={0};
for (int c=0; c<9; c++){
int d_r= arr[r][c];
if (check_r[d_r - 1]==0){
check_r[d_r - 1]=1;}
else{
printf("THE BOARD IS INVALID!!\nHINT: There's a repetition in row %d",r+1);
return 0;
}
}
}
```

LET'S MOVE TO THE BACKEND OF THE SODOKU VALIDATOR.....

Step 1) Displaying the instructions to enter input.

CODE:

```
printf ("\tSUDOKU VALIDATOR\n\n");
printf ("Instructions To Enter The Board:
\n1.The board takes input in form of rows.
\n2. Enter the entire row with each digit separated by space using spacebar
\n3. To start a new row press enter.
\n4. BE AWARE! Once you've pressed enter you cannot change the previous input.\n\nENTER THE BOARD BELOW:\n\n");
```

OUTPUT:

```
        SUDOKU VALIDATOR

Instructions To Enter The Board:
1. The board takes input in form of rows.
2. Enter the entire row with each digit separated by space using spacebar
3. To start a new row press enter.
4. BE AWARE! Once you've pressed enter you cannot change the previous input.
```

STEP 2) Taking the user's Sodoku board as input.

Code:

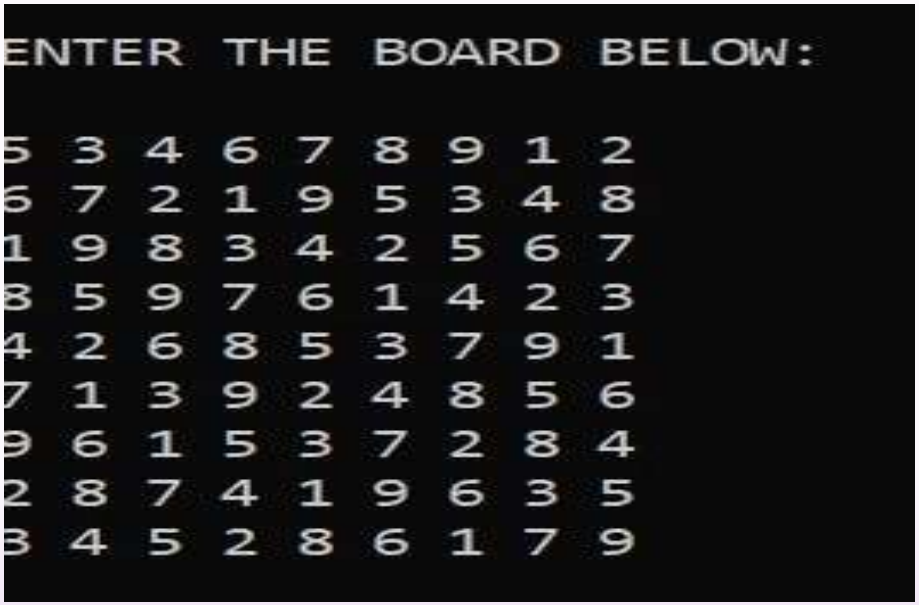
```
printf("\nENTER THE BOARD BELOW:\n\n");

int arr[9][9];

for (int r = 0; r < 9; r++)
{
    for (int c = 0; c < 9; c++)
    {
        scanf("%d", &arr[r][c]);

        if (arr[r][c] < 1 || arr[r][c] > 9)
        {
            printf("INVALID INPUT!! SUDOKU RULE VIOLATED!! Number must be between 1-9")
            return 0;
        }
    }
}
```

Output:



Output when input number is not in the range(1-9):

```
ENTER THE BOARD BELOW:

1 3 57 8 3 5 6 5
INVALID INPUT!! SUDOKU RULE VIOLATED!! Number must be between 1-9
--------------------------------
```

Step 3) Checking the rows of the entered Sodoku board to see if any element is repeating.

Code:

```
  // checking rows
for (int r=0; r<9; r++){
    int check_r[9]={0};
    for (int c=0; c<9; c++){
        int d_r= arr[r][c];
        if (check_r[d_r - 1]==0){
            check_r[d_r - 1]=1;}
        else{
            printf("THE BOARD IS INVALID!!\nHINT: There's a repetition in row %d",r+1);
            return 0;
            }
        }
    }
```

Logic behind checking repitition:

- The array check_r[9] acts as a tracker for digits 1–9 in each row.
-
- Since Sudoku digits range from 1–9, each digit can be mapped to an index in array check-r[9] using (digit-1)

- For example: if an element of a row in the Sodoku board = 5 then its index in check_r[9] array = 4.

- Each time a digit appears, the corresponding index in check_r is marked 1.

- If the same digit appears again, that index will already be 1, meaning a repetition → invalid row.

For example:

- Example (Row = 3 1 4 5 6 7 8 9 2):
- Start: check_r = {0,0,0,0,0,0,0,0,0}
- After reading 3: mark index 2 : {0,0,1,0,0,0,0,0,0}
- After reading 1: mark index 0 : {1,0,1,0,0,0,0,0,0}
- If a number repeats (say another 3 appears), then check_r[2] would already be 1, triggering invalid message.

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

RESULT:
THE BOARD IS INVALID!!
HINT: There's a repetition in row 1
```

Step 4) Checking columns of the Sodoku for repetition.

CODE:

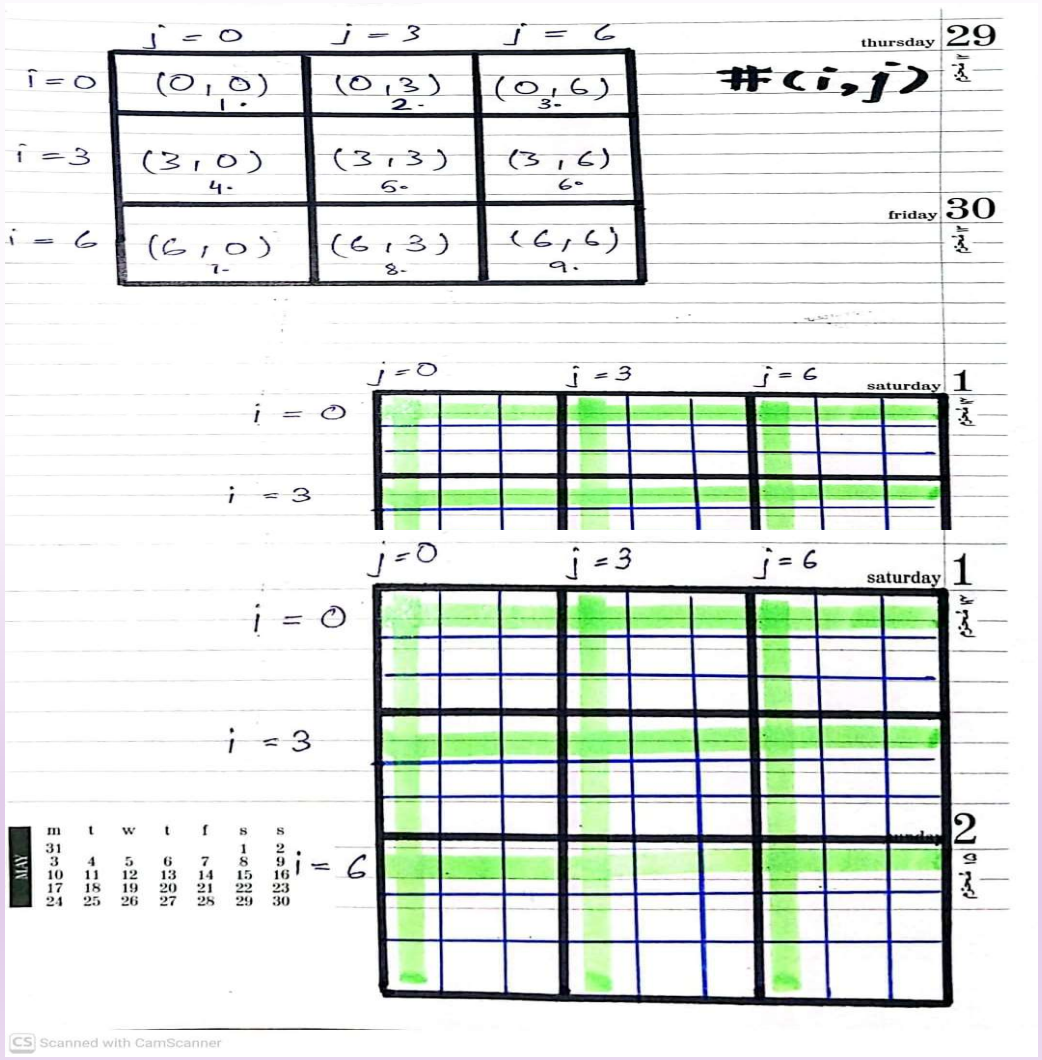
```
    // checking coloumns
for (int c=0; c<9; c++)
{
    int check_c[9]={0};
    for (int r=0; r<9; r++)
    {
        int d_c = arr[r][c];
        if (check_c[d_c - 1]==0)
        {
            check_c[d_c - 1]=1;
        }
        else
        {
            printf("THE BOARD IS INVALID!!\nHINT: There's a repetition in coloumn %d",c+1);
            return 0;
        }
    }
}
//checking sub grids
```

Code	What it does
for (int c=0; c<9; c++)	loop through each column from 0 to 8
int check_c[9] = {0};	To track the digits that we've already found once (it initially contains 0 to show that we haven't found any digit yet)
d_c - 1	Since array indices start at 0, but sudoku board starts from 1.
if (check_c[d_c - 1] == 0)	To check if we haven't spotted it yet (it being equal to 0 shows that it is being spotted for the first time)
check_c[d_c - 1] = 1;	Change the value from 0 and assign 1 to it. (to mark it as found/spotted)
else {	This means that we've already spotted that digit.
printf("THE BOARD IS check_c[d_c - 1] = 1;	Since the digit is spotted twice and sudoku rule is violated, it will display Change the value from 0 and assign 1 to it. (to mark it as found/spotted)
else {	This means that we've already spotted that digit.
printf("THE BOARD IS INVALID!!\nHINT: There's a repetition in column %d", c+1);	Since the digit is spotted twice and sudoku rule is violated, it will display a message telling the sudoku player that the board is invalid and direct them where to start correction from.
return 0; }	To terminate the program immediately after the rule is violated.

Step 5) Checking the sub-grids for repetition:

CODE:

```
//checking sub-grids
for(int i=0; i<9; i+=3){
    for (int j=0; j<9; j+=3){ //for changing grids
    int check_g[9]={0};
        for(int r=i; r<i+3; r++){
            for (int c=j; c<j+3; c++){
                int d_g=arr[r][c];
                if (check_g[d_g - 1]==0){
                    check_g[d_g - 1]=1;}
                else{
                    printf("THE BOARD IS INVALID!!\nHINT: ");
                    if (i==0)
                    {
                    if(j==0)
                    printf ("There's a repetition in sub-grid 1.");
                    else if(j==3)
                    printf ("There's a repetition in sub-grid 2.");
                    else
                    printf ("There's a repetition in sub-grid 3.");
                    }
                    else if(i==3)
                    {
                    if(j==0)
                    printf ("There's a repetition in sub-grid 4.");
                    else if(j==3)
                    printf ("There's a repetition in sub-grid 5.");
                    else
                    printf ("There's a repetition in sub-grid 6.");
                    }
                    else
                    {
                    if(j==0)
                    printf ("There's a repetition in sub-grid 7.");
                    else if(j==3)
                    printf ("There's a repetition in sub-grid 8.");
                    else
                    printf ("There's a repetition in sub-grid 9.");
                    }
```


j = 0
j = 3
j = 6
i = 0
i = 3
i = 6
(0,0) 1.
(0,3) 2.
(0,6) 3.
(3,0) 4.
(3,3) 5.
(3,6) 6.
(6,0) 7.
(6,3) 8.
(6,6) 9.
thursday 29
#(i,j)
friday 30
saturday 1
saturday 1
sunday 2
m t w t f s s
31 1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
MAY
Scanned with CamScanner

Important points:

- This tool is only a Sodoku Validator, with the objective of verifying the entered Sodoku board. And showing if it is valid or not.
- It is not necessary to enter the elements in the same manner as situated in the Sodoku board. User can enter in whatever way but using space in between is necessary.