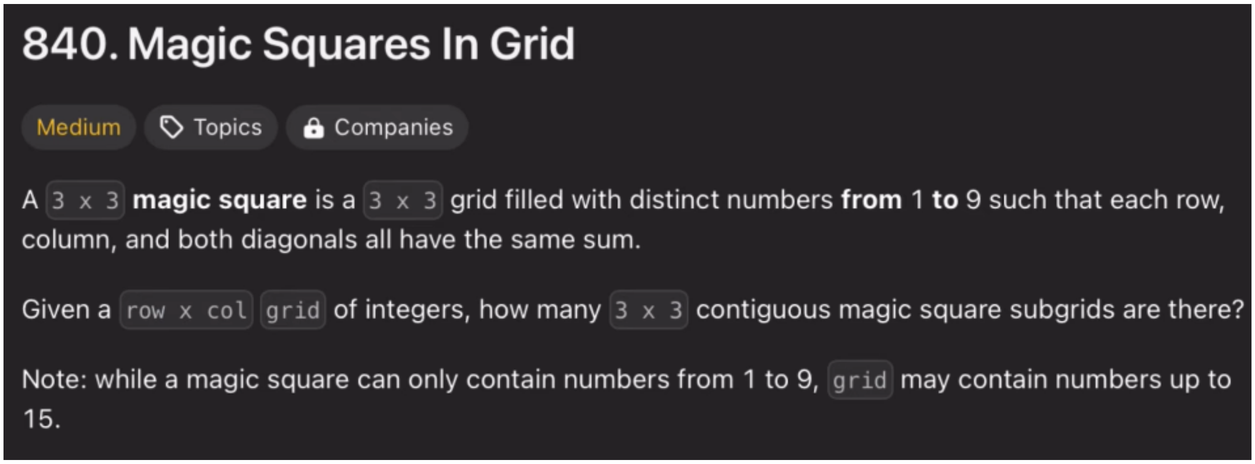
840 Magic Squares In Grid - 9/8/24 (medium)

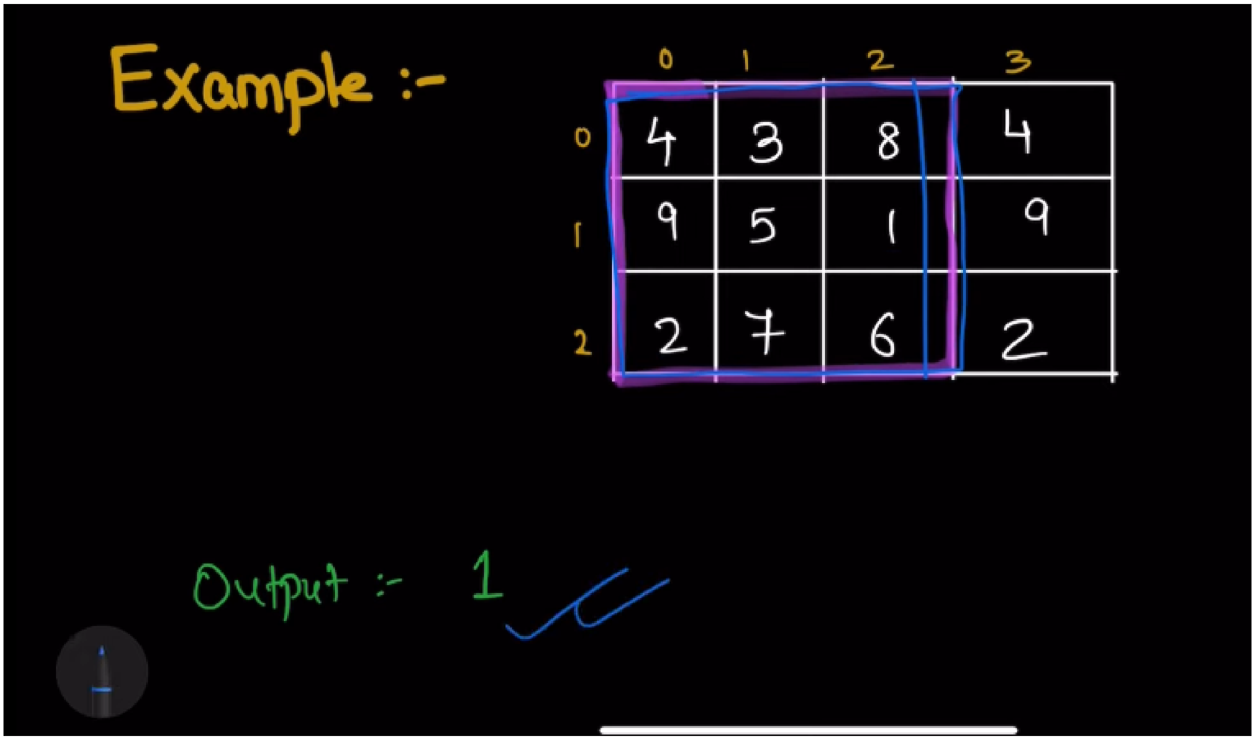


840. Magic Squares In Grid

A `3 x 3` **magic square** is a `3 x 3` grid filled with distinct numbers **from** 1 **to** 9 such that each row, column, and both diagonals all have the same sum.

Given a `row x col` `grid` of integers, how many `3 x 3` contiguous magic square subgrids are there?

Note: while a magic square can only contain numbers from 1 to 9, `grid` may contain numbers up to 15.



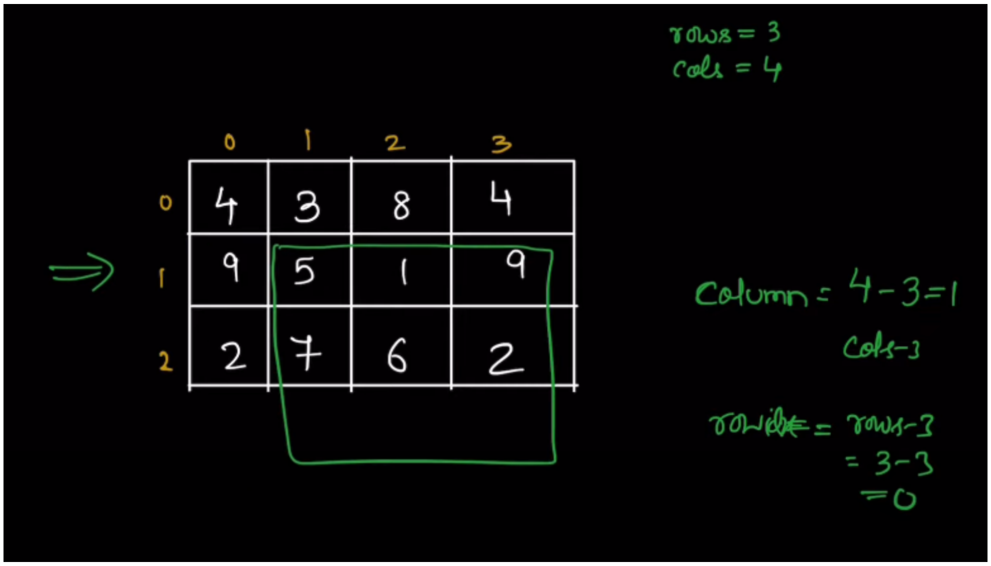
Example :-

	0	1	2	3
0	4	3	8	4
1	9	5	1	9
2	2	7	6	2

Output :- 1

3×3 magic grid is one when sum of all diagonal , rows and columsn is same then we call it magic grid.

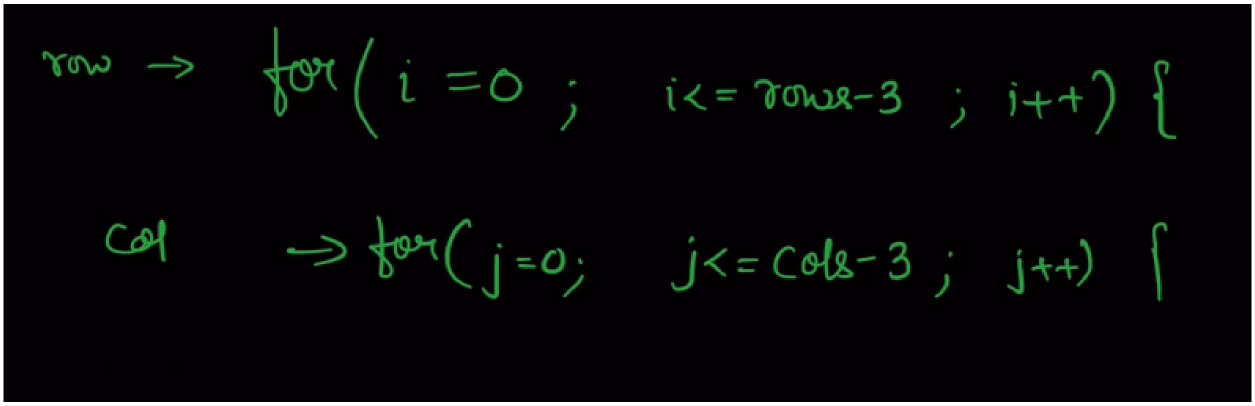
to find magic grid in any grid the formula is



	0	1	2	3
0	4	3	8	4
1	9	5	1	9
2	2	7	6	2

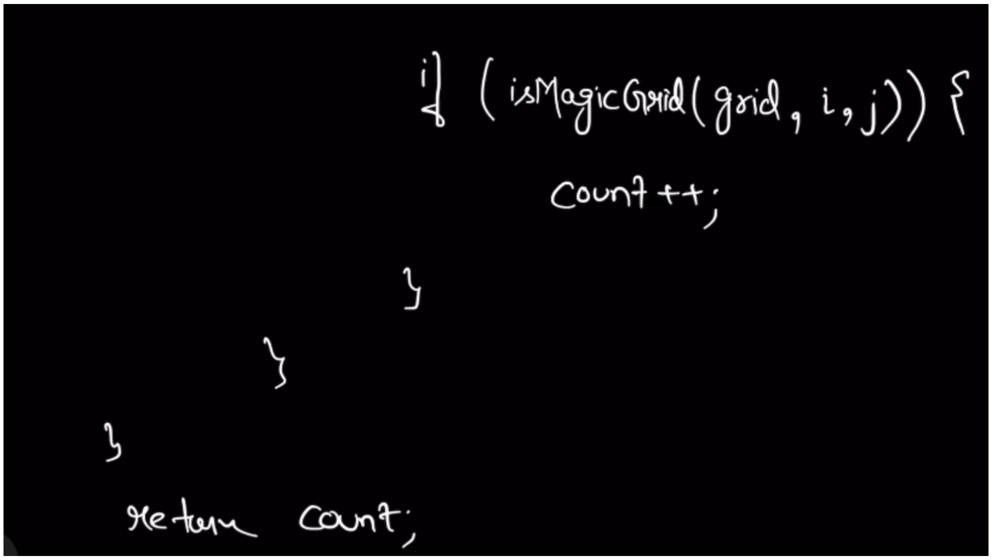
```
coulmn = cols-3
rows = rows-3 //  becuase 3x3 magic grid
```

how the code works

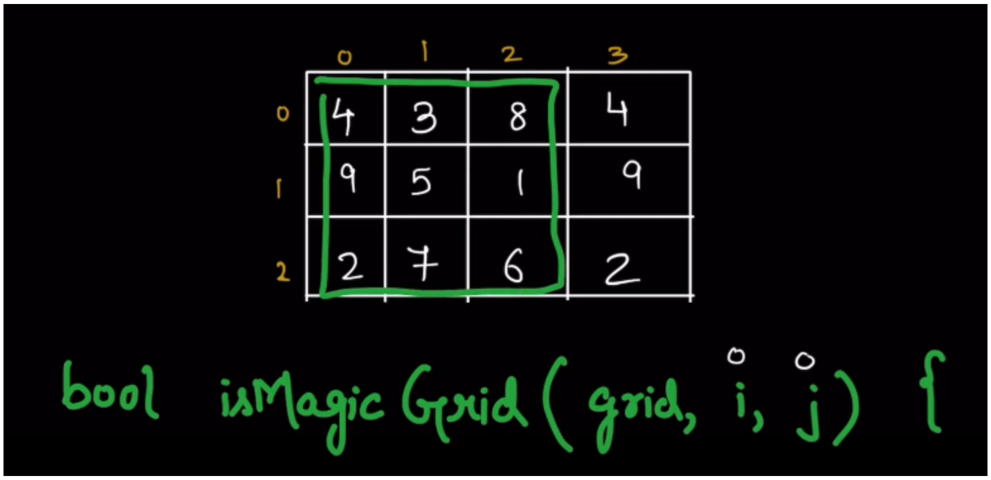


```
row → for( i =0 ;   i<= rows-3 ; i++) {

col     → for(j=0;  j<=cols-3 ;  j++)  {
```

now how to write code for IsMagicGrid

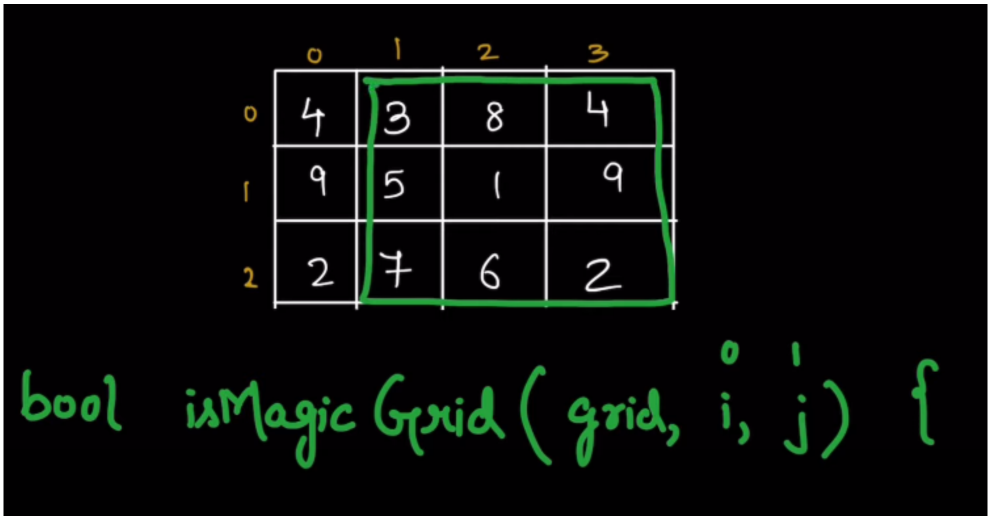


```
                    if (isMagicGrid(grid, i, j)) {
                              count++;
                    }
              }
      }
      return count;
```



	0	1	2	3
0	4	3	8	4
1	9	5	1	9
2	2	7	6	2

bool isMagicGrid(grid, $\overset{0}{i}$, $\overset{0}{j}$) {



	0	1	2	3
0	4	3	8	4
1	9	5	1	9
2	2	7	6	2

bool isMagicGrid(grid, $\overset{0}{i}$, $\overset{1}{j}$) {

3×3 must contian

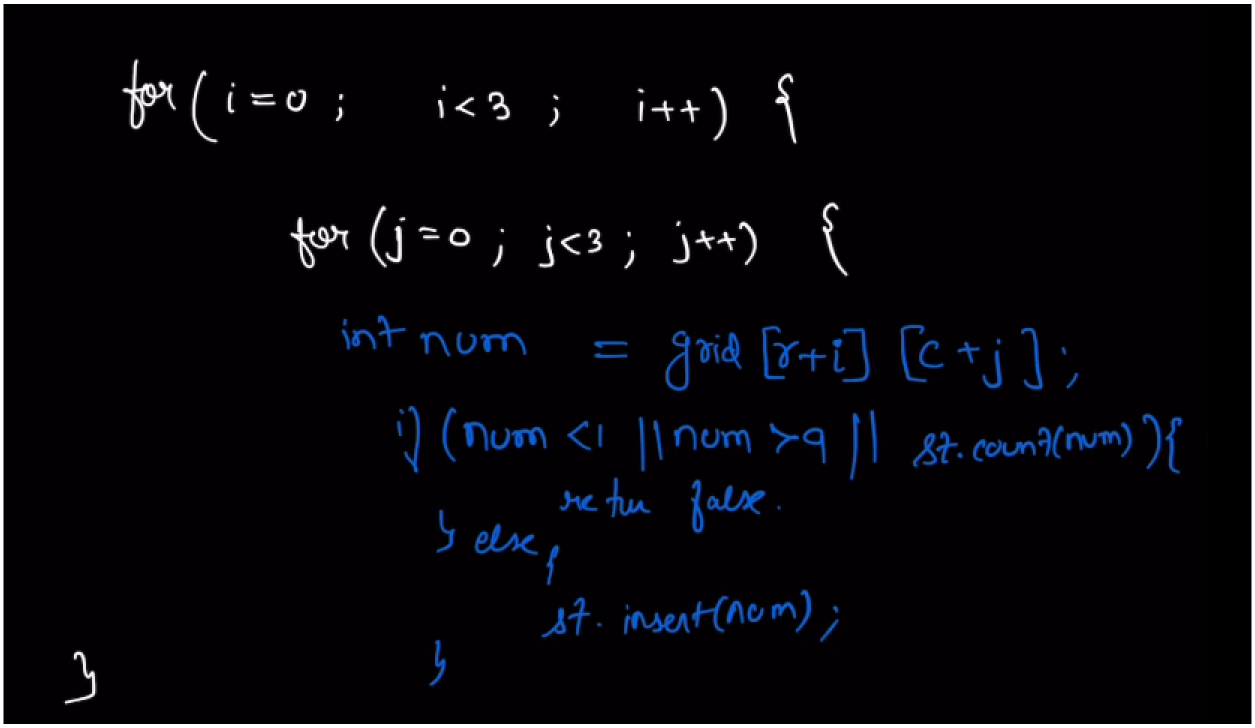
distinct 1 to 9 number

no duplicate

we will use unordered set <int> st;

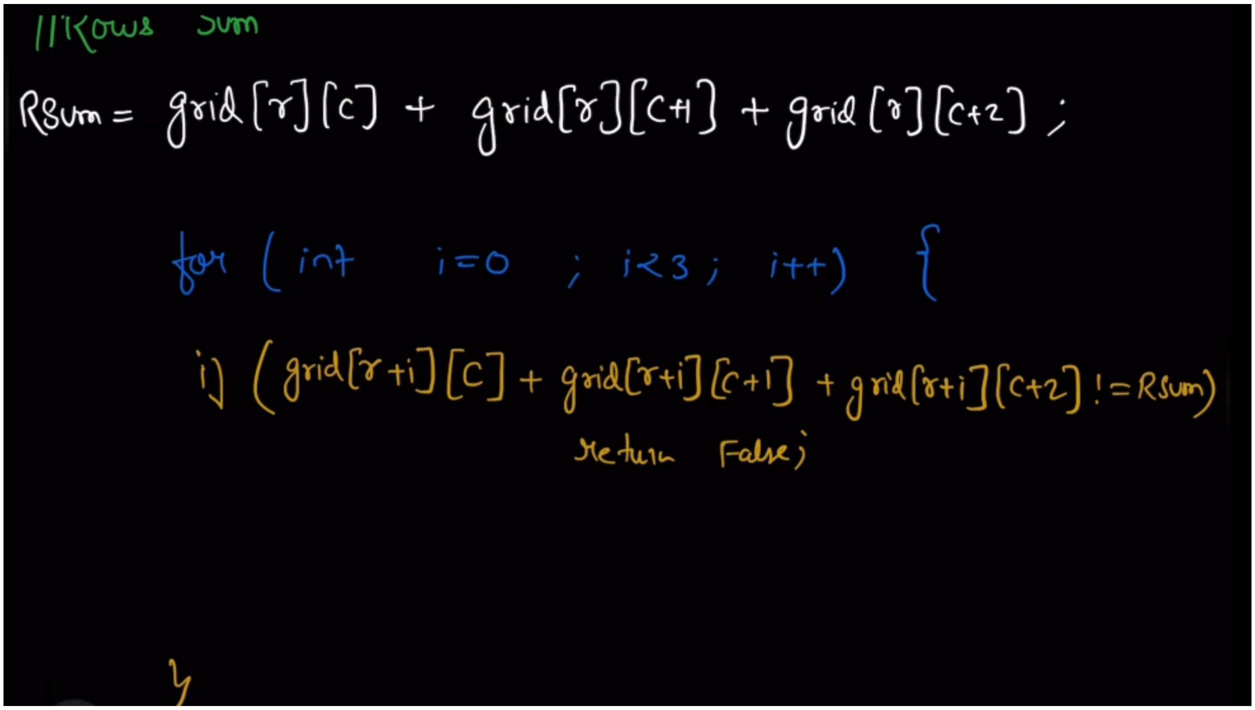
---

finding distinct number

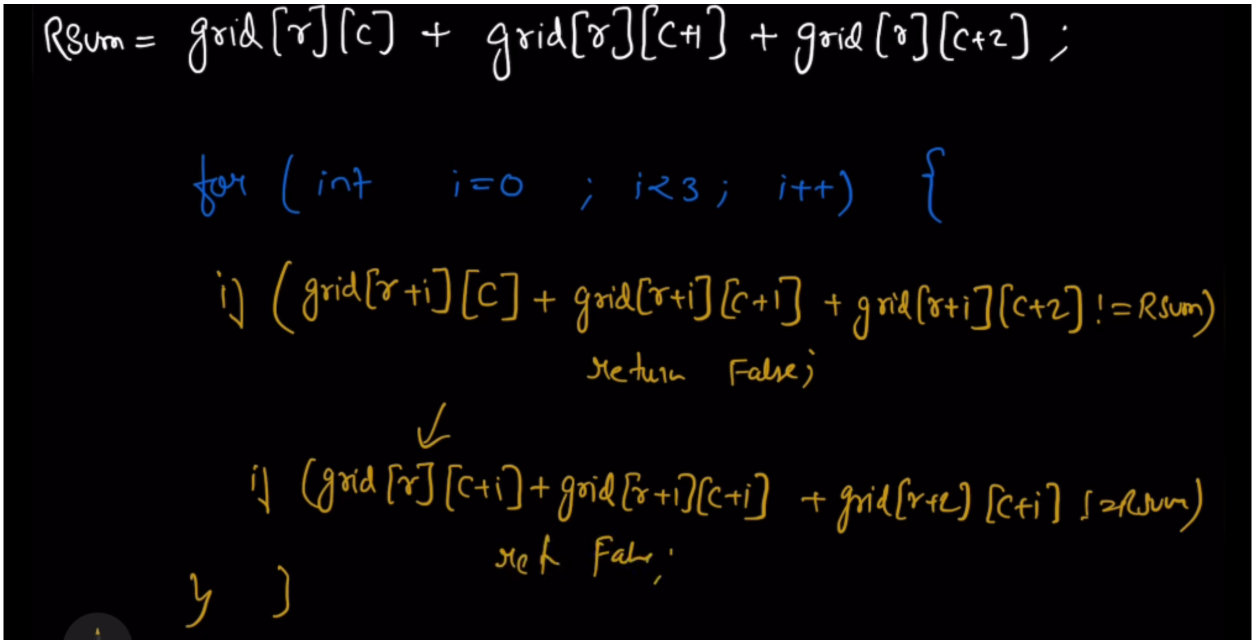


```
for ( i=0 ;    i<3 ;    i++) {

        for (j=0 ; j<3 ; j++)  {

            int num    = grid [r+i] [c+j ] ;
              if (num <1 ||num >9 || st.count(num) ){
                      retu false.
               } else {
                      st. insert(num) ;
               }
  }
```

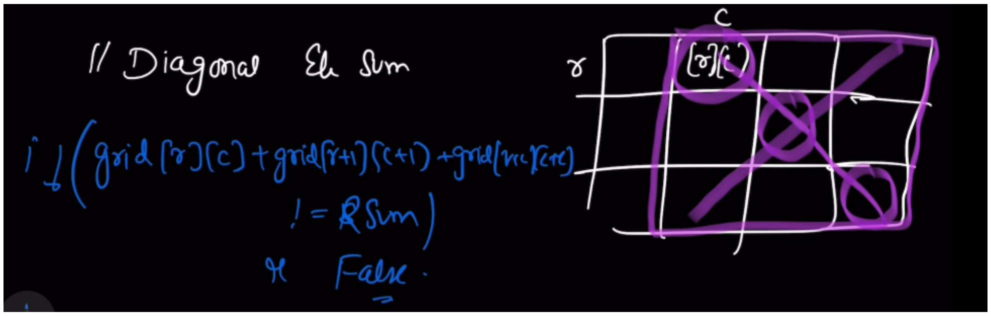
now to find sum of rows , column, and diagonal and anti diagonal



```
//Rows sum

RSum = grid[r][c] + grid[r][c+1] + grid[r][c+2] ;

    for ( int  i=0  ; i<3 ; i++)  {

      if ( grid[r+i][c] + grid[r+i][c+1] + grid[r+i][c+2] != RSum)
                    return  False;

    }
```

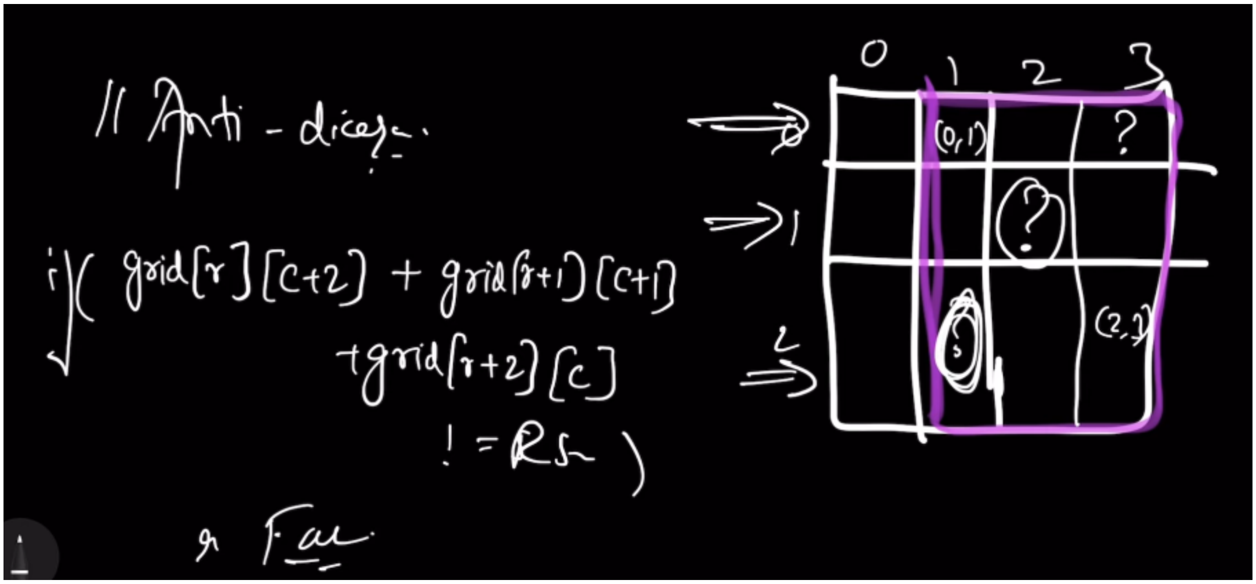


```
RSum = grid[r][c] + grid[r][c+1] + grid[r][c+2] ;

    for ( int  i=0  ; i<3 ; i++)  {

      if ( grid[r+i][c] + grid[r+i][c+1] + grid[r+i][c+2] != RSum)
                    return  False;
                 ↙
        if (grid[r][c+i] + grid[r+1][c+i] + grid[r+2][c+i] != RSum)
                  ret  False;
    }  }
```



// Diagonal Ele Sum

if (grid[r][c] + grid[r+1][c+1] + grid[r+2][c+2]
!= RSum)
rt False.



// Anti - diag.

if(grid[r][c+2] + grid[r+1][c+1]
+grid[r+2][c]
!= RSm)
rt Fal.

code:

```
class Solution {
public:

    bool isMagicGrid(vector<vector<int>>& grid,int r,int c){
        unordered_set<int> st;
        for(int i=0;i<3;i++){
            for(int j =0 ; j<3 ; j++){
                int num = grid[r+i][c+j];
                if(num<1||num>9||st.count(num)){
                    return false;
                }
                else{
                    st.insert(num);
                }
            }
        }

        int sum = grid[r][c]+grid[r][c+1]+grid[r][c+2];
        for(int i=0;i<3;i++){

            //rows
            if(grid[r+i][c]+grid[r+i][c+1]+grid[r+i][c+2]!=sum)
                return false;
            }

            //cols
            if(grid[r][c+i]+grid[r+1][c+i]+grid[r+2][c+i]!=sum)
                return false;
            }

        }

        //diagonal
        if(grid[r][c]+grid[r+1][c+1]+grid[r+2][c+2]!=sum){
            return false;

        }
        //Antidiagonal
        if(grid[r][c+2]+grid[r+1][c+1]+grid[r+2][c]!=sum){
            return false;

        }
        return true;
```

```
    }

    int numMagicSquaresInside(vector<vector<int>>& grid) {
        int rows = grid.size();
        int cols = grid.size();
        int count = 0;
        for(int i=0;i<=rows-3;i++){
            for(int j=0;j<=cols-3;j++){
                if(isMagicGrid(grid,i,j)){
                    count++;
                }
            }
        }
        return count;
    }
};
```