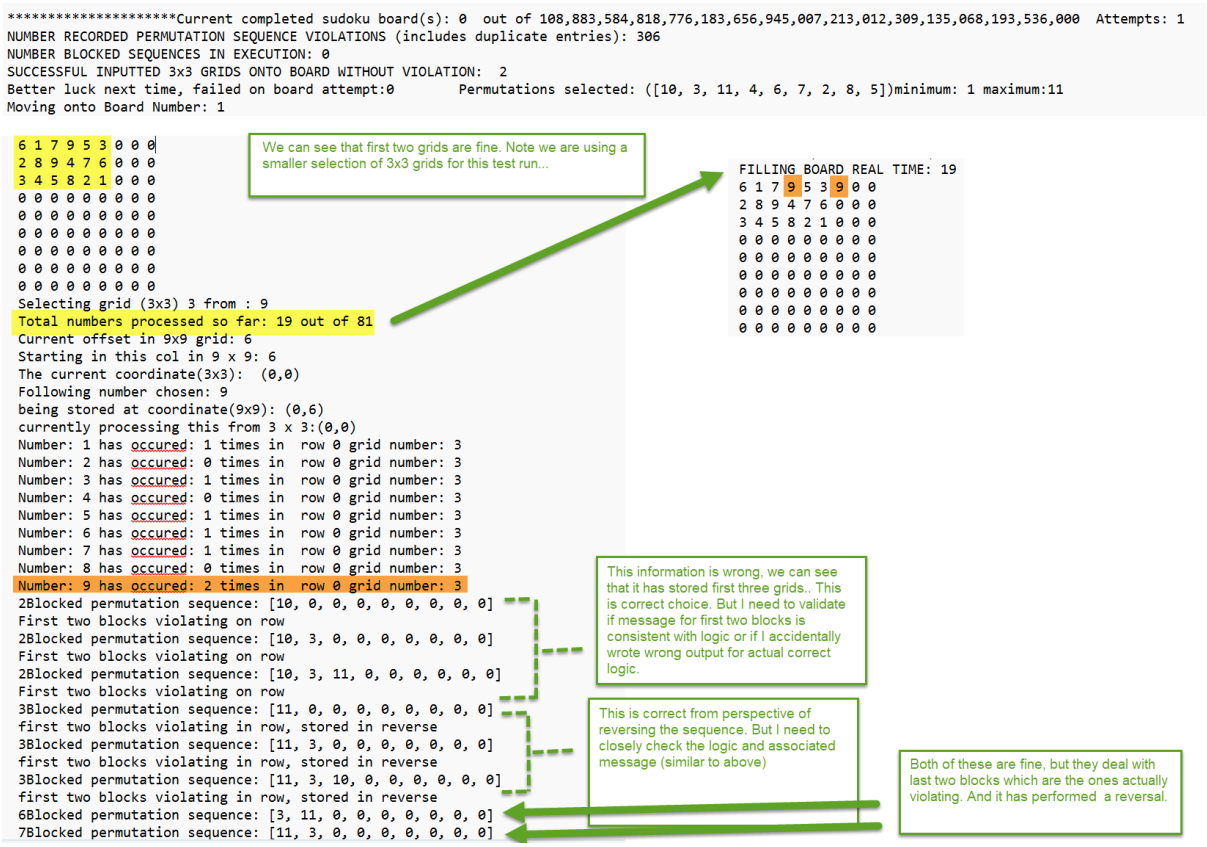
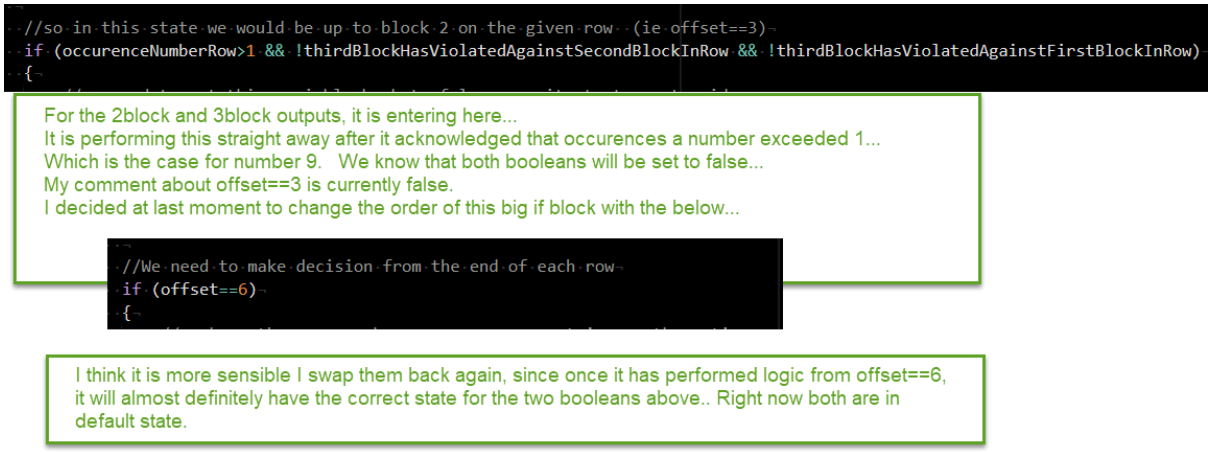
TEST CASE: Executing the code inline with revision from row perspective:



I am now going back over my code again.

I am not entirely sure but once I completed the new coding, I swapped order of these major blocks of code:



TEST CASE: Rotated the code blocks

```
**********Current completed sudoku board(s): 0  out of 108,883,584,818,776,183,656,945,007,213,012,309,135,068,193,536,000  Attempts: 1
NUMBER RECORDED PERMUTATION SEQUENCE VIOLATIONS (includes duplicate entries): 372
NUMBER BLOCKED SEQUENCES IN EXECUTION: 0
SUCCESSFUL INPUTTED 3x3 GRIDS ONTO BOARD WITHOUT VIOLATION:  1
Better luck next time, failed on board attempt:0        Permutations selected: ([7, 2, 9, 5, 6, 1, 8, 4, 10])minimum: 1 maximum:11
```

```
2Blocked permutation sequence: [7, 0, 0, 0, 0, 0, 0, 0, 0]
First two blocks violating on row
2Blocked permutation sequence: [7, 2, 0, 0, 0, 0, 0, 0, 0]
First two blocks violating on row
3Blocked permutation sequence: [2, 0, 0, 0, 0, 0, 0, 0, 0]
first two blocks violating in row, stored in reverse
3Blocked permutation sequence: [2, 7, 0, 0, 0, 0, 0, 0, 0]
first two blocks violating in row, stored in reverse
Number: 9 has occured: 0 times in  row 0 grid number: 2
FILLING BOARD REAL TIME: 10
8 4 1 8 0 0 0 0 0
7 2 3 0 0 0 0 0 0
9 5 6 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
```

IT now seems perfect, any I do not expect anymore block messages until it reaches last grid in the row..
This was another modification amongst the mass changes in the code.

TEST CASE: Analysing further into screen outputs

```
Number: 4 has occured: 2 times in  row 0 grid number: 2
2Blocked permutation sequence: [7, 0, 0, 0, 0, 0, 0, 0, 0]
First two blocks violating on row
2Blocked permutation sequence: [7, 2, 0, 0, 0, 0, 0, 0, 0]
First two blocks violating on row
3Blocked permutation sequence: [2, 0, 0, 0, 0, 0, 0, 0, 0]
first two blocks violating in row, stored in reverse
3Blocked permutation sequence: [2, 7, 0, 0, 0, 0, 0, 0, 0]
first two blocks violating in row, stored in reverse
Number: 5 has occured: 1 times in  row 0 grid number: 2
Number: 6 has occured: 0 times in  row 0 grid number: 2
Number: 7 has occured: 0 times in  row 0 grid number: 2
Number: 8 has occured: 2 times in  row 0 grid number: 2
2Blocked permutation sequence: [7, 0, 0, 0, 0, 0, 0, 0, 0]
First two blocks violating on row
2Blocked permutation sequence: [7, 2, 0, 0, 0, 0, 0, 0, 0]
First two blocks violating on row
3Blocked permutation sequence: [2, 0, 0, 0, 0, 0, 0, 0, 0]
first two blocks violating in row, stored in reverse
3Blocked permutation sequence: [2, 7, 0, 0, 0, 0, 0, 0, 0]
first two blocks violating in row, stored in reverse
Number: 9 has occured: 0 times in  row 0 grid number: 2
FILLING BOARD REAL TIME: 12
8 4 1 8 5 4 0 0 0
7 2 3 0 0 0 0 0 0
9 5 6 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
```

For the same grid 2, it has performed same activities again. Although it is correct on the observations, we have already stored [7,2] and [2,7] once the 8 appeared on

```
//so in this state we would be up to block 2 on the given row  (ie offset==3)
if (occurenceNumberRow>1 && !thirdBlockHasViolatedAgainstSecondBlockInRow && !thirdBlockHasViolatedAgainstFirstBlockInRow)
{
    //we need to set this variable back to false once it starts next grid
    //need to find best location in the code where this occurs
    hasRegisteredViolationBlock=true;
```

It does not appear that I kept the condition for into the if statement

I will now try again with this

```
//so in this state we would be up to block 2 on the given row  (ie offset==3)
if (occurenceNumberRow>1 && !thirdBlockHasViolatedAgainstSecondBlockInRow && !thirdBlockHasViolatedAgainstFirstBlockInRow
&& !hasRegisteredViolationBlock)
{
```

TEST CASE: Running test again

```
Following number chosen: 9
being stored at coordinate(9x9): (2,5)
currently processing this from 3 x 3:(2,2)
Number: 1 has occured: 0 times in  row 2 grid number: 2
Number: 2 has occured: 1 times in  row 2 grid number: 2
Number: 3 has occured: 1 times in  row 2 grid number: 2
Number: 4 has occured: 0 times in  row 2 grid number: 2
Number: 5 has occured: 1 times in  row 2 grid number: 2
Number: 6 has occured: 0 times in  row 2 grid number: 2
Number: 7 has occured: 0 times in  row 2 grid number: 2
Number: 8 has occured: 2 times in  row 2 grid number: 2
Number: 9 has occured: 1 times in  row 2 grid number: 2
FILLING BOARD REAL TIME: 18
5 1 6 6 7 1 0 0 0
7 4 9 2 4 3 0 0 0
2 3 8 8 5 9 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
```

It is perfect execution, it has reached to end of 18 entries and stored violation of the first and second block at single moment...

I will now follow outputs beyond this since its critical area of code:

TEST CASE: Examining further outputs

There seemed to be lots repeating, so I tightened up some of the if loops:

```
//in practice both thirdBlockHasViolatedAgainstSecondBlockInRow and thirdBlockHasViolatedAgainstFirstBlockInRow should be tru
if (occurenceNumberRow==3 && thirdBlockHasViolatedAgainstFirstBlockInRow && thirdBlockHasViolatedAgainstSecondBlockInRow)
    {
        thirdBlockHasViolatedAgainstSecondBlockInRow=false;
        thirdBlockHasViolatedAgainstFirstBlockInRow=false;
```

```
//if it has not performed a break, we know that other Occurrence of
//possibleNumbers[j] is in gridNumber 2.
//so we store gridNumber 2 and 3
else
{
    if (!thirdBlockHasViolatedAgainstSecondBlockInRow)
    {
    thirdBlockHasViolatedAgainstSecondBlockInRow=true;

    //we are storing grid 2 and 3 as blocked
    blockedPermutationNumberSequence[position][0]=storeRetrieved3x3Grid[1];
```

```
    if (numberOf3x3Processed==9)
    {
        colCount=0;
        rowCount=0;

        //I have kept this here
    hasRegisteredViolationBlock=false;
    }
```

This variable reset was best placed here since it marked end of filling the 3x3 grid

```
Number: 9 has occured: 6 times in  row 2 grid number: 2
FILLING BOARD REAL TIME: 18
7 4 6 2 4 9 0 0 0
8 5 9 8 6 3 0 0 0
1 3 2 7 5 1 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
Selecting grid (3x3) 3 from : 9
Total numbers processed so far: 19 out of 81
Current offset in 9x9 grid: 6
Starting in this col in 9 x 9: 6
The current coordinate(3x3):  (0,0)
Following number chosen: 2
being stored at coordinate(9x9): (0,6)
currently processing this from 3 x 3:(0,0)
Number: 1 has occured: 0 times in  row 0 grid number: 3
Number: 2 has occured: 2 times in  row 0 grid number: 3
6Blocked permutation sequence: [1, 9, 0, 0, 0, 0, 0, 0, 0]
Blocks 2 and 3 violating in row
7Blocked permutation sequence: [9, 1, 0, 0, 0, 0, 0, 0, 0]
Blocks 2 and 3 violating in row, storing in reverse
Number: 3 has occured: 0 times in  row 0 grid number: 3
Number: 4 has occured: 2 times in  row 0 grid number: 3
4Blocked permutation sequence: [11, 9, 0, 0, 0, 0, 0, 0, 0]
Blocks 1 and 3 violating in row
5Blocked permutation sequence: [9, 11, 0, 0, 0, 0, 0, 0, 0]
```

This is ok

Perhaps it is simple sensible to maximise the for loop or perform a break if suitable

It seems issues are spiralling, so it must be related to variables being tangled up.