22nd Annual Rowan University Programming Contest

hosted by the
Computer Science Department

Friday, 25 April 2008

Contest Problem

Rowan University™

1 Background

A **Latin square** is a square grid filled with symbols in such a way that each symbol occurs exactly once in each row and column.

Grid #1

1	2	3
3	1	2
2	3	1

Grid #2

A	C	G	T
G	T	A	C
C	A	G	T
T	G	C	A

Sudoku puzzles are similar to Latin squares, but the grid is divided into sub-squares called 'Blocks', and duplicates are not allowed in Blocks. As usually published, the grid is 9×9 with 3×3 Blocks. Some squares are pre-filled, and the solver completes the grid by putting the digits 1-9 into the empty squares without improper duplication.

Obviously, any solved Sudoku is also a valid Latin square, but the inverse is not true.

Grid # 3 — Sudoku

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

Sudoku-X adds the restriction that duplicates are also not allowed on the major diagonals.

Obviously, any solved Sudoku-X is both a solved Sudoku and a valid Latin square, but (as above) not all solved Sudokus qualify as Sudoku-X.

Grid #4 — Sudoku-X

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

Note that because Sudoku and Sudoku-X require their Blocks to be square, the number of rows/columns must be a perfect square. A Latin square can be any size (5×5, 12×12, &c.), but Sudoku/Sudoku-X must be 4×4, 9×9, 16×16, and on up through 25, 36, 49, &c. It wouldn't be possible to construct a 5×5 Sudoku, because there would be no way to create square Blocks of size 5.

The challenge for this contest is to write a program which analyses a square grid of numbers and classifies it as exactly as possible.

2 Terminology

Rows go across. **Columns** go up-and-down. Numbering starts in the **upper-left**.

Col. 1 ⇓ Col. 4 ⇓

Row 1⇒	α			
				β
Row 4⇒			π	

α is in Row 1, Column 1.
β is in Row 2, Column 4.
π is in Row 4, Column 3.

Grid coordinate pairs have the row first:
α is at (1,1); β is at (2,4); π is at (4,3).

Blocks in Sudoku and Sudoku-X grids will be numbered **left-to-right**, **top-to-bottom**:

For a 9×9 Sudoku:

A								**B**
							C	
					D			

Block Layout:

1	2	3
4	5	6
7	8	9

A is in Block 1.
B is in Block 3.
C is in Block 6.
D is in Block 8.

For a 4×4 Sudoku:

X			
			Y
		Z	

Block Layout:

1	2
3	4

X is in Block 1.
Y is in Block 2.
Z is in Block 4.

Note that a $\mathcal{N} \times \mathcal{N}$ Sudoku will have $\mathcal{N}$ Blocks, and each Block will have $\mathcal{N}$ spaces.

Diagonals for Sudoku-X grids will be numbered as follows:

1								2
	1						2	
		1				2		
			1		2			
				X				
			2		1			
		2				1		
	2						1	
2								1

1			2
	1	2	
	2	1	
2			1

In the case of a grid with an odd number of rows/columns, the center square will be in both diagonals.

In the case of a grid with an even number of rows/columns, no square will be in both diagonals.

3 Input

3.1 Input Specification

For text input, your program should accept input in the following format:

1. An integer, $\mathcal{G}$, where $1 \leq \mathcal{G} \leq 50$, which is the number of grids in a given input file.
2. $\mathcal{G}$ data sets, each of which is in this format:
   (a) An integer, $\mathcal{S}$, where $1 \leq \mathcal{S} \leq 50$, giving the rows and columns in this grid.
   (b) $\mathcal{S}$ lines, each of which will have $\mathcal{S}$ integers, giving the puzzle grid. To keep the input simple, the input will follow the standard Sudoku convention of using the numbers 1–$\mathcal{S}$. An unfilled space will be represented by zero.

3.2 Sample Input #1

Data in file	Item # above:	Meaning in plain English
`2`	1	*this file has 2 data sets*
`9`	2a	*grid 1 is 9×9*
`2 5 0 0 3 0 9 0 1`		
`0 1 0 0 0 4 0 0 0`		
`4 0 7 0 0 0 2 0 8`		
`0 0 5 2 0 0 0 0 0`		
`0 0 0 0 9 8 1 0 0`	2b	*the 81 values in grid 1*
`0 4 0 0 0 3 0 0 0`		
`0 0 0 3 6 0 0 7 0`		
`0 7 0 0 0 0 0 2 3`		
`9 0 3 0 0 0 6 0 4`		
`5`	2a	*grid 2 is 5×5*
`2 5 4 1 3`		
`3 1 5 4 2`		
`4 3 2 5 1`	2b	*the 25 values in grid 2*
`1 4 3 2 5`		
`5 2 1 3 4`		

(This input will be available on the website as **sample1.txt**.)

NOTES:

You may choose to have your program read the input from the keyboard, or ask the user for a filename and then read the file. Users of GUI-based programming environments may prefer to use text boxes into which the values can be entered, and buttons to begin their calculation. Any reasonable variation in the spirit of the problem is acceptable.

4 Output

4.1 Output Specification

For each data set configuration, your program must generate output as follows:

1. The text '`Analyzing G Grids`', where **G** is the number of grids in the input.
2. The text '`Grid N:`', where **N** is the number of the grid being reported on.
3. The text '`Size: S x S`', where **S** is the number of rows and columns in this grid.
4. One or more of the following lines, depending on which are correct:

Solved Sudoku-X	completely filled in per Sudoku-X rules.
Unsolved Sudoku-X	some blanks; no improper duplicates per Sudoku-X rules.
Incorrect Sudoku-X	incorrect duplicates were found.
Solved Sudoku	completely filled in per Sudoku rules.
Unsolved Sudoku	some blanks; no improper duplicates per Sudoku rules.
Incorrect Sudoku	incorrect duplicates were found.
Solved Latin Square	completely filled in per Latin square rules.
Unsolved Latin Square	some blanks; no duplicates per Latin square rules.
Incorrect Latin Square	incorrect duplicates were found.

5. If improper duplicates are found, print the duplicated symbol and *all* the grid coordinates it appears in (as described in §2 on page 3).

A solved or unsolved Sudoku-X does not need to be analyzed further. An incorrect Sudoku-X which is a solved or unsolved Sudoku does not need to be analyzed as a Latin square. (Note that a size such as 8×8 can *only* be analyzed as a Latin square.)

4.2 Sample Output #1

`Output Generated`
`by Program`
	`Item #`
	`above:`
		`Meaning`
		`in plain English`

`Analyzing 2 Grids`	1	*there are two grids in the file*
` Grid 1:`	2	*this is for grid 1*
`  Size:  9 x 9`	3	*grid 1 is 9×9*
`  Incorrect Sudoku-X`	4	*grid 1 has improper duplicates*
`  2 is repeated on diagonal 1`	5	*reporting a duplicated symbol;* **notes:**
`    (1,1) (4,4) (8,8)`	5	*1. all occurrences listed on one line*
`  9 is repeated on diagonal 2`	5	*2. duplicates must be listed in increasing*
`    (5,5) (9,1)`	5	*order by row and then column*
`  Unsolved Sudoku`	4	*by Sudoku rules, the grid has no improper duplicates*
` Grid 2:`	2	*this is for grid 2*
`  Size:  5 x 5`	3	*grid 2 is 5×5*
`  Solved Latin Square`	4	*grid 2 is completely filled in per Latin square rules*

(This output corresponds to the sample input from §3.2.)

5 Sample Data

5.1 Sample Input #2

```
5                              this file has 5 data sets
9                              grid 1 is 9×9
2 5 0 0 3 0 9 0 1
0 1 2 0 0 4 0 0 0
4 0 7 0 0 0 2 0 8
0 0 5 2 0 0 0 0 0
0 0 0 0 9 8 1 0 0              the values in grid 1
0 4 0 0 0 3 0 0 0
0 0 0 3 6 0 0 7 2
0 7 0 0 0 0 0 0 3
9 0 3 0 0 0 6 0 4
4                              grid 2 is 4×4
1 2 3 4
3 4 1 2                        the values in grid 2
2 1 4 3
4 3 2 1
9                              grid 3 is 9×9
2 5 0 0 3 0 9 0 1
0 1 0 0 0 4 0 0 0
4 0 7 0 0 0 2 0 8
0 0 5 2 0 0 0 0 0
0 0 0 0 9 8 1 0 0              the values in grid 3
0 4 0 0 0 3 0 0 0
0 0 0 3 6 0 0 7 2
0 7 0 0 0 0 0 0 3
9 0 3 0 2 0 6 0 4
6                              grid 4 is 6×6
1 0 3 4 5 6
3 3 0 0 3 2
2 0 1 5 4 3                    the values in grid 4
0 4 6 3 2 1
5 1 4 6 3 0
0 4 0 3 0 5
9                              grid 5 is 9×9
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              the values in grid 5
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
```

5.2 Sample Output #2

```
Analyzing 5 Grids
 Grid 1:
   Size: 9 x 9
   Incorrect Sudoku-X
     2 is repeated on diagonal 1
       (1,1) (4,4)
     9 is repeated on diagonal 2
       (5,5) (9,1)
     2 is duplicated in block 1
       (1,1) (2,3)
   Incorrect Sudoku
     2 is duplicated in block 1
       (1,1) (2,3)
   Unsolved Latin square
 Grid 2:
   Size: 4 x 4
   Incorrect Sudoku-X
   1 is repeated on diagonal 1
     (1,1) (4,4)
   4 is repeated on diagonal 1
     (2,2) (3,3)
   1 is repeated on diagonal 2
     (2,3) (3,2)
   4 is repeated on diagonal 2
     (1,4) (4,1)
   Solved Sudoku
 Grid 3:
   Size: 9 x 9
   Incorrect Sudoku-X
     2 is repeated on diagonal 1
       (1,1) (4,4)
     9 is repeated on diagonal 2
       (5,5) (9,1)
   Unsolved Sudoku
 Grid 4:
   Size: 6 x 6
   Incorrect Latin square
     4 is repeated in column 2
       (4,2) (6,2)
     3 is repeated in column 4
       (4,4) (6,4)
     3 is repeated in column 5
       (2,5) (5,5)
     3 is repeated in row 2
       (2,1) (2,2) (2,5)
 Grid 5:
   Size: 9 x 9
   Unsolved Sudoku-X
```

5.3 Notes On Sample Input/Output #2

All Grids: Every improper duplicate is printed out for each class of grid. Each duplicate is documented by listing all the places where it occurs.

All Grids: Since the digits in a Sudoku are not processed numerically, they serve only as unique symbols. The reasoning would be the same if one used the letters A—I, or the planetary symbols (☿♀♁♂♃♄♅♆♇), or any other symbols. To keep matters simple, especially when handling larger grids such as 16×16, all data sets will use **integers only**. Further, the integers in each grid will be consecutive from 1 through the size of the grid, inclusive.

All Grids: Your output **does not** have to match the sample output exactly as regards letter case or precise spacing. It should be neat and readable, but does *not* have to match exactly.

Grid 1: This grid does not qualify as a Sudoku-X, and must be reported on as a Sudoku. It also does not qualify as a Sudoku, and must be reported on as a Latin square. Since the duplicates which disqualify it from the more restrictive categories are not a problem for a Latin square, it passes that test.

Grid 2: This grid fails the Sudoku-X test, but passes the Sudoku test. Since it is completely filled in, it is considered solved.

Grid 3: A Sudoku may be impossible in a way that isn't immediately apparent from simply scanning for duplicates. Here is Grid 3 displayed with guidelines:

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

Here, the solver has placed a **2** in the center of the bottom row. That does not introduce a duplicate in the bottom-middle block, in the bottom row, or in the center column. But it does make the puzzle unsolvable, because there is now nowhere to place a two in the top-center block. The only available square already has a 4 in it.

Your program will ***NOT*** have to handle this case.

Your program only has to scan for duplicates in the diagonals, rows, columns, and blocks. Therefore, **Grid 3** should is considered an unsolved Sudoku, even though it cannot actually be solved from its current state.

Grid 4: This grid cannot be a Sudoku or Sudoku-X, because the number of rows/columns is not a perfect square. Therefore, it is analyzed only as a Latin square, and no report on its status as Sudoku or Sudoku-X is generated. Note that **all** repeats of 3 in row 2 are listed **on one line**.

Grid 5: This grid is completely blank, and thus has no duplicates at all.

6 Test Data

Run your program on this input and print the results. **You must submit printed output to earn full points.** Your program will also be run on data known only to the judges.

6.1 Test Input #1

```
4
25
 23  8  2 18  6 13 19 25  7 10 20 11 12 16 24  9 22 17 14 15  5  4  3  1 21
 24 17  3 22  5  2 11 18 14 20 15  7 23 25  9  1  4 21 13 12 10 16  8 19  6
  4 10 19 13 16  6  1 17  9 12  2 18  8 21 22  3  5 20 25 11 24  7 14 23 15
  7 20 25 21 15 23 24 16 22  5  4  1  3 19 14  6 10 18  8  2 13 12 11  9 17
 14 11 12  9  1  8 21 15  4  3  6 13  5 17 10 23 24 16 19  7 20 25 22  2 18
 13 21  5 16 12 20  9 22  8  6 11 19 14 24 17 10  2 23 15  4  7  3 25 18  1
  6 23 10 17 19 16 12  1 18 13 21  4 22 20  8 25  7 24  5  3 11  9  2 15 14
  9 15 11  4  8 14  7 24  2 17  1  3 18 13 25 12 19 22  6 20 16 23 21 10  5
 22  1  7 24  3 25  5 11 10 15 23 16  6 12  2 17 21  9 18 14  8 19 13  4 20
  2 18 14 20 25 21  3 23 19  4  5 10  9 15  7  8 11 13  1 16  6 24 12 17 22
 25 19 13  7  9 17 18 20  6 16  3 23 11  5  1 14  8  2  4 22 21 15 24 12 10
 18 22 16 14 17 10  4  9 21  8 12 15 24  6 20  5  1  3 23 19 25  2  7 11 13
 21 24 23  8 11  7 15 14  1 19 22  2 10  9 13 20 12 25 16 17  4 18  5  6  3
  1  3  4  5 10 24  2 12 11 22 25 17  7 14 16 13  6 15 21 18 19 20  9  8 23
 15 12  6  2 20  5 23  3 13 25  8 21 19  4 18  7  9 11 24 10 22 17  1 14 16
 20  5 15  1 23 11 25  4 24  7 10  8 13 18  3  2 16 19  9  6 14 21 17 22 12
 12  7 22  6 14  1 16 21  3 18  9 20  2 11 19 24 15  5 17 13 23 10  4 25  8
 10  4  9 19 21 15 13  2 17 23 16 14 25 22  5 18 20  1 12  8  3 11  6 24  7
 16 13  8  3  2 22  6 10 12  9 17 24  4 23 15 11 25 14  7 21  1  5 18 20 19
 17 25 24 11 18 19 20  8  5 14  7 12 21  1  6  4  3 10 22 23  9 13 15 16  2
 11 16  1 12 24 18  8  5 20 21 14  9 15  3 23 19 13  6  2 25 17 22 10  7  4
 19 14 20 25 22 12 10 13 15 11 18  6  1  7 21 16 17  4  3  9  2  8 23  5 24
  8  9 21 23  4  3 22  7 16  2 24  5 17 10 11 15 14 12 20  1 18  6 19 13 25
  3  2 17 15 13  9 14  6 23 24 19 25 20  8  4 22 18  7 10  5 12  1 16 21 11
  5  6 18 10  7  4 17 19 25  1 13 22 16  2 12 21 23  8 11 24 15 14 20  3  9
36
 36 15  0 23  0 16  0  2 13  0  0  0 29  0  0  0  0 27  0  0 14  0  0 10  0  0  0  0  0 25 19  0  6  1  0  0
  0  0  0 28  0  0  0  0  0  0  0 16  1  0  0 34 10  0 21  0  0 12  6  0 30  0  0 31  0  3 23  0 25  4  0  2
 20  0  0 11  9 29 32 30  0  0  5 33 28  0 14  3  0  0  0 16 27  0  8  0 12  0  0  7  0  0  0 15  0 35 36  0
  0 32 14 21 13 35 29  4  3 31  0  0  7 22 19  0  0 36 18  0 30  0 15  5  0 10 23 20  9  0  0  0  0 34 12  0
 10  0  0 27  2  0  0  0  0 25 23  0  0  0 20  0 35 12  0  0  7  3 34  0 32 36 19  0  0  0 11  0 18 31  9  0
 30  0 19 12  0  0 21  6  0  0  0  0  0  0 26  0 31  0 29 17  0  0  9 13  0 33 18  2 15 27 16  0  3  0  8 14
  7  0  2  0 25 19  0 24 18  0  1  0  0  0 29 20  0  0  0  0 35 32  0 11  0  0  0 23  4 30  0 26 14 16  0  0
  0  0  0 20 12 11  0  0  6  0  0 10  4 33  0  0  0  0  0  0 29  0  0 18  0 25  0  0  0 19  0 22  0  0  0 28
  0 13  0 10 26  8 25  0  0  0 16  9 22  0 30 32  0  0 28 24 20  5  0  0 29 18 31 17 36  0  0 12  2  3  0  0
  0  0  3 24 34  0 30 20  2  0  0  0  0 26  0 14  0  0  7  9 19  0 27 21  0  0 15  0  6  0 29  0  0 25 13 10
 22  0  0  4 30  0  0  3  0 13 14 29 36 27  0 15  0  0 12 23  0  0  0  8 24  0 34  0  0  0 21  0  9  0  0  0
 28 31  1  0 29  0  0 26  0  4  0 22 10 19  0 12  2  0 13  0 34  0  0  0  0  0  0  0  0 35  0  0  0 33  0  0
 35 16  0  3  0  4  0  0  1  0  0  0  0  0  9  8 13 26  0  0 12 18  0  0  0  0 11 10  0  2  0  0  0 28 14  0
  0 23  0  0  0 13  0 27 28 29 35 21  6  0 25  5 20  0  0 11  0 22  7  0  0  0 36  0 34  0  0  9 24 15 16 32
  0  0  9  0 11 22  0 13  7  0  0  2  0  0  0  0 17  0  0  0  0 31 24  1 21  0 30 35 28  0  0 10  0 20  0  5
 19  8 28 32  0 17  0 12  9  0  4 23  0 24  0  7 22 18 14  0  0  0  0  2  0  0 16  1 27  0  0  0 11  0 21  0
  1  0  0  0 15  0  0  0  0  3  8 36  0  0  4 11  0  0 17  0 16  9  0  0 33  0  7 19  0  0  0  0  0 27  0  0
  0  0  0  0  0 18  0  0 24  0 22  0 32  2  0 36 19 16  0  0 28  0  0  0  0  0 29 15 20  9  0  0 17  0  0 30
 29  0  0 25  0  0 10 33  8  7  0  0  0  0  0  6  0  0 30  1 18  0  4 14  0 24  0  9  0  0  3  0  0  0  0  0
  0  0 34  0  0  0  0  0 30 16  0 14  0  0 10 25  0 31  0  0  9 23  0  0 18 22  1  0  0  0  0 21  0  0  0 33
  0 30  0 26  0  0  0 22 23 28  0  0 24  0  0  0  0 33  2 31  8  0 16  0  5 35  0 29 11  0 12  0  1  9 17 18
 13  0 36  0 23  0  0 35 21 24  0  5 12  1  2  0  0  0  0 22  0  0  0  0 14  0  0 30  8  0 31  7  0  6  0  0
 18 11 21  1 19  0  0 36  0  9  0  0  0  8 16  0 28  0  0 25 15  7  0 24 31 23 10  6  3  0 35  0  0  0 30  0
  0 27 24  0  0  0 19  0 15 18  0  0  0  0 23 29  0  0 36 21  3 26  0  0  0  0  0 33  0  0 10  0 32  0 22 11
  0  0 13  0  0  0  8  0  0  0  0  0  0  0  0 22  0 20  0  3 24  0  1 16  4  0 32  0 21  0  0  5  0  2 34  9
  0  0  0 30  0 23  0  0  0 21  0  6  2  0  0  0 12 24  0  0 26  0 28 19 16  9 14  0 10  0  0  8 33  0  0  3
  5 29 10  0  0 32  0  9  0 12  0  0 21  6  0  4 27 13  0  0 22  0 20  0  0  0  0 11 33  1  0 18 31 26  0  0
  0  0 18  2 36  0  0 29  4 14 20 28  0  0  3 10 33  8  0  0 32 25  0 15  7 31  0  0  0 23 17 24 21  0 11  0
 12  0  0  0  1  0 17  0  0  0  7  0 26  0  0 18  0  0  0  0  0  0 14 31 20  0  0 36  0  0 27  4 13  0  0  0
  0  0 17 31 33  0 13  5 27  0  0  0 23  0 32  1  0  0  0  0  2 21  0  0  0 15  0 34 25  0 14 20  0 12  0 16
 32 26  0 35  0 34 31 18 22 20 10  0 13 11  0  0 29 19  0 33  0 24  0  0  0  0  0  0 23  8  0  0  4 21  0 15
  0  9  5 33  0 27  0  0  0  1 28 19  0 25 17  2  0  0 34 10  0 36  0  0  0 11 20  0  0  0  0  6 30  0  0 24
  0 12 11  0  0  0  0 23  5  6 33  0 31  3  0 35  0  7  1  0  0 14 17 29  0  0  4 25 13 16  9 36 10 19 26  0
  0  6 23  0 18  0  0  0 26  0  0 25  0  4  0 27 15  0  0  0 11 30  0  9  1  7  0  0 29 10 33 13 12  0  0 22
 14  0  4 29  0 25  9  0 17  0  0  3  0 30 33  0  0 10  0 27  5  0  0 26 19  0  0  0  0  0  0  0 16  0  0  0
  0  0 16 13  0 10  4  0  0  0  0  0 34  0  0 28  0  0 15  0  0  0  0 25  0  0  0 32 17  0  5  0 29  0  3  1
9
 1 1 1 1 1 1 1 1 1
 1 1 0 0 0 0 0 1 1
 1 0 2 0 0 0 2 0 1
 1 0 0 2 2 2 0 0 1
 1 0 0 2 2 2 0 0 1
 1 0 0 2 2 2 0 0 1
 1 0 2 0 0 0 2 0 1
 1 1 0 0 0 0 0 1 1
 1 1 1 1 1 1 1 1 1
3
 1 2 3
 2 3 1
 3 1 2
```