Practice Arena

Practice problems aimed to improve your coding skills.

- PRACTICE-02_SCAN-PRINT
- PRACTICE-03_TYPES
- LAB-PRAC-02_SCAN-PRINT
- LAB-PRAC-01
- PRACTICE-04_COND
- BONUS-PRAC-02
- LAB-PRAC-03_TYPES
- PRACTICE-05_COND-LOOPS
- LAB-PRAC-04_COND
- LAB-PRAC-05_CONDLOOPS
- PRACTICE-07_LOOPS-ARR
- LAB-PRAC-06_LOOPS
- LAB-PRAC-07_LOOPS-ARR
- LABEXAM-PRAC-01_MIDSEM
- PRACTICE-09_PTR-MAT
- LAB-PRAC-08_ARR-STR
- PRACTICE-10_MAT-FUN
- LAB-PRAC-09_PTR-MAT
  - Mr C writes a Story
  - Matrix Arithmetic
  - Spin the Matrix
  - Crony Capitalization
  - Matrix Mirroring
  - Sodoku
  - The Last Line
  - Singular Value Decomposition
  - Matrix Flip
  - Now we are in Rome
  - Search for the Submatrix
  - Convoluted Convolutions
- LAB-PRAC-10_MAT-FUN
- PRACTICE-11_FUN-PTR
- LAB-PRAC-11_FUN-PTR
- LAB-PRAC-12_FUN-STRUC
- LABEXAM-PRAC-02_ENDSEM
- LAB-PRAC-13_STRUC-NUM
- LAB-PRAC-14_SORT-MISC

Sodoku

LAB-PRAC-09_PTR-MAT

Sodoku [20 marks]

Problem Statement
The Sodoku is a popular puzzle, commonly played in its 9 x 9 version. We will look at a generalization of this puzzle. An n x n Sodoku is defined whenever n is a perfect square like 4 or 9 or 16 etc and is presented as an n x n matrix, each of whose entries is an integer between 1 and n.

Let k be the square root of n. Then for an n x n matrix, we define n non-overlapping *boxes* as follows. Consider the following example 4 x 4 Sodoku.
1 2 3 4
3 4 1 2
4 3 2 1
2 1 4 3

Here, the 4 boxes are
BOX 1
1 2
3 4

BOX 2
3 4
1 2

BOX 3
4 3
2 1

BOX 4
2 1
4 3

i.e. the boxes are divided as

1 2 || 3 4
3 4 || 1 2
=========
4 3 || 2 1
2 1 || 4 3

Note that boxes are non-overlapping and are numbered left to right and top to bottom. There are exactly n boxes in the matrix. A Sodoku is considered valid if

1. Every row of the matrix has all numbers from 1 to n occurring exactly once.
2. Every column of the matrix has all numbers from 1 to n occurring exactly once.
3. Every box in the matrix has all numbers from 1 to n occurring exactly once.

You will be given n as a strictly positive perfect square integer in the first line of the input. Then you

will be given the n rows of this matrix, each row on a separate line, with a single space separating two elements of a row. If the given Sodoku is valid, simply print "Valid Sudoku" (without quotes) in the output and that is it.

However, if the Sodoku is not valid, you have to first described which all rows are invalid (in increasing order of rows) then describe which all columns are invalid (in increasing order of columns) then describe which all boxes are invalid (in increasing order of boxes).

Caution

1. We will not penalize you for trailing new lines at the end of your output. However, do not have trailing spaces at the end of any line of your output.

Code to manipulate matrices

```
int m, n;
scanf("%d %d", &m, &n);
int num[m][n], i, j;
for(i = 0; i < m; i++)
      for(j = 0; j < n; j++)
            scanf("%d", &num[i][j]);
            printf("%d", num[i][j]);
```

HINT: You may use the sqrt function by including math.h to calculate the square root of a number.

EXAMPLE 1:

INPUT
4
1 2 3 4
3 4 1 2
4 3 2 1
2 1 4 3

OUTPUT:
Valid Sudoku

EXAMPLE 2:

INPUT
4
1 2 3 2
3 4 1 2
4 3 2 1
2 1 4 3

OUTPUT:
Row 1 is invalid
Column 4 is invalid
Box 2 is invalid

Grading Scheme:

Total marks: **[20 Points]**

There will be partial grading in this question. There will be several lines in your output. Printing each line correctly, in the correct order, carries equal weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

Start Solving! (/editor/practice/6187)