

Lab Assignment 9 - 03/26/2009

Name: _____

Section: 8:00 11:00 12:30 6:30

Review Problem: Latin Squares

A *Latin square* is an $n \times n$ matrix filled with numbers $1, 2, \dots, n$ so that each row and each column has no duplicate numbers. (Example: A Sudoku is a 9×9 Latin square with the additional requirement that the “boxes” have no duplicate values) Your goal today is to write a program that checks a Latin square of any size. Write a program in source file `latin.c` that performs the following operations.

Milestone 1: Input a number n from the user, and use it to dynamically allocate a two-dimensional `int` array of size n by n .

Milestone 2: Input n^2 numbers from the user, filling the matrix, first by rows and then by columns.

Milestone 3: Check that each row has each of the numbers $1, 2, \dots, n$. If a row does not, then output “FAIL: ROW #”, where # is replaced by the failing row index, and quit (`return 0`);).

Milestone 4: Check that each column has each of the numbers $1, 2, \dots, n$. If a column does not, then output “FAIL: COLUMN #”, where # is replaced by the failing column index, and quit (`return 0`);).

Milestone 5: After checking the rows and columns, if the program is still running the matrix is a Latin square. Output “PASS”.

Compile your code into a program, `latin`, and show the instructor. If you are unable to finish, the last milestone reached will be marked.

You will receive full credit for the lab in two conditions:

1. you complete all milestones, or
2. you spend the entire lab making progress on the assignment.

You will receive an extra credit point for completing the lab in class and/or an extra credit point for completing the lab before 5:00 P.M. March 31 (if you do not complete it today, turn in `latin.c` by email). **You must turn in this sheet today!**

Expected Behavior:

Input	5 1 2 3 4 5 5 3 4 1 2 3 4 2 5 1 2 5 1 3 4 4 1 5 2 3	3 1 2 3 2 3 3 3 1 2	4 1 2 3 4 2 3 4 1 3 4 1 2 4 1 3 2
Output	PASS	FAIL: ROW 1	FAIL: COLUMN 2

Extra Credit Problem: Write a program in a source file `magic.c` to take the same input, but instead check that the $n \times n$ matrix is a *magic square*, with output being PASS or FAIL.

See http://en.wikipedia.org/wiki/Magic_square for more information. Submit your solution by email before 5:00 P.M., Tuesday March 31 for the credit.