COP 3503 Programming Fundamentals for CIS Majors II, Spring 2017

Programming Assignment 1

Out: Jan. 27 (Friday), 2017

Due: 7:25am on Feb. 13 (Monday), 2017

Problem Description

Write a C++ program that creates four(4) different N*N magic squares. A square matrix is the arrangement of the numbers 1, 2, ..., N^2, in which the sum of rows, columns, and diagonals are the same. The users (i.e., TAs) will specify the size of the square matrix: N. The value N must be an odd number between 3 and 15.

Example Run

For example, you program is expected to run in the following way:

```
INPUT>> Enter the size of a magic square: 3

OUTPUT>> Magic Square #1 is:

   2   7   6
   9   5   1
   4   3   8

OUTPUT>> Checking the sums of every row:      15 15 15
OUTPUT>> Checking the sums of every column:   15 15 15
OUTPUT>> Checking the sums of every diagnoal: 15 15

OUTPUT>> Magic Square #2 is:

   8   3   4
   1   5   9
   6   7   2

OUTPUT>> Checking the sums of every row:      15 15 15
OUTPUT>> Checking the sums of every column:   15 15 15
OUTPUT>> Checking the sums of every diagnoal: 15 15
```

```
OUTPUT>> Magic Square #3 is:

   8   1   6
   3   5   7
   4   9   2

OUTPUT>> Checking the sums of every row:      15 15 15
OUTPUT>> Checking the sums of every column:   15 15 15
OUTPUT>> Checking the sums of every diagnoal: 15 15

OUTPUT>> Magic Square #4 is:

   2   9   4
   7   5   3
   6   1   8

OUTPUT>> Checking the sums of every row:      15 15 15
OUTPUT>> Checking the sums of every column:   15 15 15
OUTPUT>> Checking the sums of every diagnoal: 15 15
```

Hints to find magic squares

1. (Data Structures:) You should use arrays to generate and store the integers. You should practice C++ modeling to create classes and methods to compile the program, then run the executable code.

2. (Algorithms:) Many good heuristics do exist to find a magic square quickly. You should be able to find them via Google search on Internet. However, finding ALL magic squares requires careful examination of all permutation of all these N*N numbers (e.g., 256! for a 16*16 matrix). When the matrix size N is big enough, the computation for ALL magic squares will become intensive.

3. (Computation:) You are only required to find four (4) magic squares for the full credits since we just want you to get familiar with C++ programming via this assignment. However, you are encouraged to explore the computational complexity of finding ALL magic squares, if time allows.

4. There are many ways to model/code in C++. Thus, there may not be a standard solution. You need to explain well on your modeling and coding with comments in the pa1.h and pa1.cpp file(s).

5. Please do not take user's input N, then declare the size of arrays. It is not a good practice since malicious users may input a very large number (e.g., 1,000,000).

6. Note that squares cannot be hard-coded, but must actually be computed at runtime.

Submission Guilde-lines

1. You must finish this assignment with your individual effort. Your C++ source code file MUST be named as "pa1.cpp" and C++ header/class file should be named as "pa1.h".

2. You are allowed to use any IDE. However, please test your program via g++ compiler (i.e., g++ -Wall) on the CISE machines to make sure it runs correctly. Note the "-std c++11" option is acceptable by TAs. We recommend you to start the testing 48 hours before the submission deadline.

3. Please upload the source code file(s) via the CANVAS system as the attachment(s). Please submit the source code file(s) ONLY. NO need to compress the source code file(s) if the size is small.

4. Make sure you submit your assignment BEFORE the deadline. We recommend the submission to be completed 30 minutes before the deadline. Late submission will NOT be accepted by CANVAS system !!

Grading Criteria

1. Successful Compilation (30%): Your source code should be able to compile using "g++ -Wall" command without any error or warning. The output should be a valid executable. See lab tutorial for the commends. Please note that we will be using g++ compiler on Linux to grade your programs. If you are using other compilers or IDE (e.g., Visual C++), it is recommended that you test the source codes with g++ 48 hours before the CANVAS submission (i.e., to make sure there is no warning). Note again the "-std c++11" option is acceptable by TAs.

2. Program Correctness (40%): The executable should be able to run correctly by giving out the required output. Be sure to test all cases (e.g., 3,5,7,9,11,13 and 15) for the proper outputs and correctness. Un-expected results may occur when the matrix size is large.

3. Programming Style (30%): Good coding style is a key to efficient programming. We encourage you to write clear and readable codes. You should adopt a sensible set of coding conventions, including proper indention, necessary comments and more. Here are some

guidelines of good programming style: http://en.wikipedia.org/wiki/Programming_style

Final Notes

1. Remember to start the programming assignments as soon as possible. Unlike the conventional assignments, programming assignments sometimes take un-predictable amount of time to finish. Thus, have the code running first, then polish it later with the extra time before the deadline.

2. Remember you should always write your own code and never copy-and-paste from other students' work or other sources. There are indeed many tools (like Stanford Moss) to detect the code similarity.

3. Programming assignments are usually designed by TAs. If you have any question or concern, please feel free to contact TAs. Our goal is to let you experience the fundamentals of computer science. If you like the programming experience, you are one of us !!

4. HAPPY CODING ... and GOOD LUCK !!