Write the Verifier.h(the class definition for the Verifier class)

And Verifier.cpp(method definitions) in Linux.

The makefile, the Sudoku.cpp and the solutionFile that contains 7 solution txt provided below should all be in the same directory as Verifier.h and Verifier.cpp.

Sample output is provided at the end of this Word document. Please write some comment so that I can learn which step is having what kind of function.

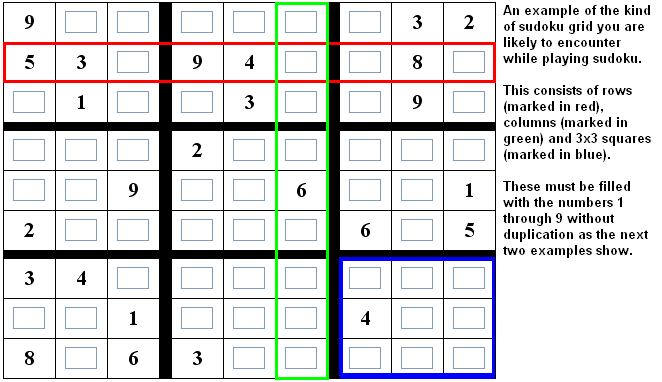
After it’s running correctly in Lunix, download the Verifier.h and the Verifier.cpp into your local file and upload.

Extra credit part is also required.

**Overview**

*This assignment covers two-dimensional arrays.*

[Sudoku](http://en.wikipedia.org/wiki/Sudoku) is a logic-based, combinatorial number-placement puzzle. The rules of Sudoku are quite simple. In a traditional sudoku puzzle (and there are variations) the objective is to fill a 9x9 grid with digits so that each column, each row, and each of the nine 3x3 sub-grids that compose the grid (also called "boxes", "blocks", "regions", or "sub-squares") contains all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which typically has a unique solution.

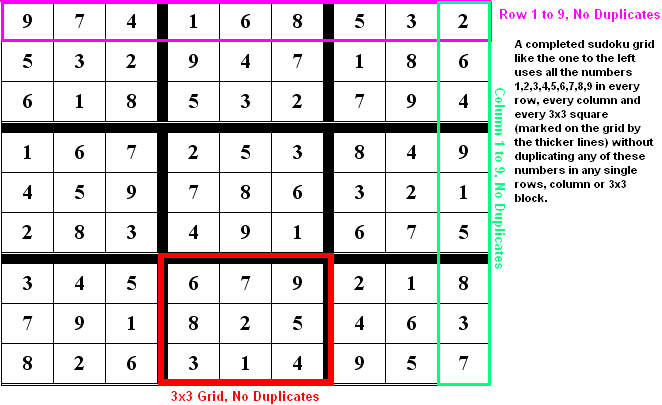
*An example of a Sudoku puzzle*

In order for a solution to a Sudoku puzzle to be correct, the following conditions must hold true:

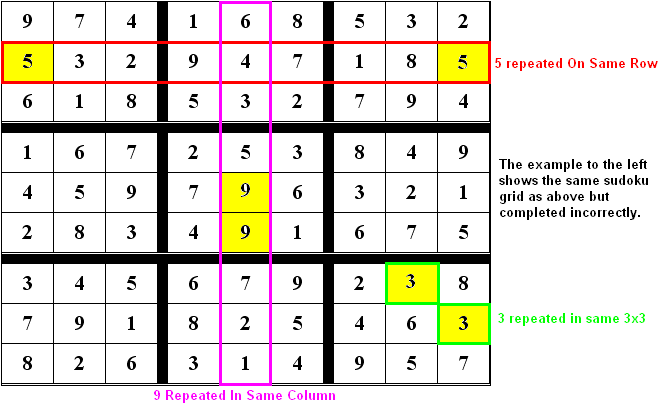
1. Every row should contain the digits 1 to 9 but should not repeat the digits 1 to 9 at any point within that row.
2. Every column should contain the digits 1 to 9 but should not repeat the digits 1 to 9 at any point within that column.
3. Every 3x3 sub-grid should contain the numbers 1 to 9 but should not repeat the numbers 1 to 9 at any point within that sub-grid.

Here are some examples of puzzle solutions:

*A correctly filled Sudoku grid*



*An incorrectly filled Sudoku grid*



Writing a [computer program to solve Sudoku puzzles](http://en.wikipedia.org/wiki/Sudoku_algorithms) can be quite challenging, and typically uses algorithms and data structures outside the scope of an intermediate class in programming. However, writing a program that will **check** a solution to a Sudoku puzzle and verify that it is correct is a lot easier.

**Program**

Write a class called Verifier that can be used to verify whether a Sudoku puzzle solution is correct (i.e., whether or not it meets the conditions outlined above).

As with Assignment 2, the class definition for the Verifier class should be placed in a header file (Verifier.h) while the method definitions should be placed in their own source code file (Verifier.cpp).

*Data members*

The Verifier class should have a private data member to represent a Sudoku grid as a two-dimensional array of 9 rows, each with 9 columns. The exact data type of the array elements is up to you; you might decide to store the digits of the Sudoku grid as integers, or you may choose to store them as characters.

You may create other private data members for the class if you want to.

*Methods*

The Verifier class should have the following public methods. You are welcome to create additional private methods for the class if you want to.

* Verifier default constructor - This constructor has no parameters. It should set all of the elements of the grid array to 0.
* readGrid() - This method takes one parameter: a pointer to a constant character (data type const char\*), which will point to an array of characters that contains the name of a file to use as input. It returns nothing. The method should read the contents of the input file into the elements of the grid array.

An input file contains exactly 81 numbers, arranged in 9 rows of 9 columns each, separated by whitespace. For example:

2 3 4 9 5 6 8 1 7

9 5 7 8 1 4 2 6 3

1 8 6 3 7 2 4 5 9

5 4 9 6 8 1 7 3 2

6 1 8 7 2 3 5 9 4

7 2 3 4 9 5 6 8 1

3 9 2 5 6 7 1 4 8

4 7 5 1 3 8 9 2 6

8 6 1 2 4 9 3 7 5

* printGrid() - This method takes no arguments and returns nothing. It should print the Sudoku grid array to the screen as 9 rows of 9 columns (the same way the grid appears in the input file). For example:
* 2 3 4 9 5 6 8 1 7
* 9 5 7 8 1 4 2 6 3
* 1 8 6 3 7 2 4 5 9
* 5 4 9 6 8 1 7 3 2
* 6 1 8 7 2 3 5 9 4
* 7 2 3 4 9 5 6 8 1
* 3 9 2 5 6 7 1 4 8
* 4 7 5 1 3 8 9 2 6
* 8 6 1 2 4 9 3 7 5
* verifySolution() - This method takes no arguments. It should return a Boolean value - true if the Sudoku grid array contains a valid solution, false if not.

**Driver Program**

A driver program, sudoku.cpp, is provided for this assignment. The purpose of a driver program is to test other pieces that you code. You do not need to write the driver program yourself..

//PROGRAM: sudoku.cpp=>This program tests the functionality of the //Verifier class.

#include <iostream>

#include <string>

#include "Verifier.h"

using std::cout;

using std::endl;

using std::string;

#define NUM\_FILES 7

int main()

{

Verifier v;

string fileName;

cout << "Sudoku Verifier\n";

for (int i = 1; i <= NUM\_FILES; i++)

{

cout << endl;

// Construct file pathname

fileName = string("/home/CS215/Data/semester/solutionFile/solution")

/\*=>the highlighted part is an example of how you link the solution txt files.\*/

+ (char)('0' + i) + ".txt";

// Read the solution file as input

v.readGrid(fileName.c\_str());

// Print the Sudoku grid

v.printGrid();

// Verify whether or not the solution is correct

if (v.verifySolution())

cout << "\nThis is a valid Sudoku solution\n";

else

cout << "\nThis is not a valid Sudoku solution\n";

}

return 0;

}

**Implementation Hints**

The driver program should not be modified for your final submission. But while you're developing, modifying the driver program can definitely be in your best interest. For example, you may want to take out the loop and concentrate on working with a single input file.

The input files used by the driver program:



The files are named solution1.txt through solution7.txt. You can run the driver program on Unix without copying these files to your home directory, but you are welcome to make copies of them, modify the copies for testing purposes, download them to your local computer, etc. You can also easily create your own data files for testing.

The exact technique you use to verify whether a solution is correct or not is up to you, but it will most likely require you to loop through the columns of each row looking for duplicate values, loop through the rows of each column looking for duplicate values, and loop through the rows and columns of each sub-grid looking for duplicate values.

**Makefile**

A makefile is required for this assignment.

A sample makefile is given below.

#

# PROGRAM: makefile

#

# Compiler variables

CCFLAGS = -Wall -std=c++11

# Rule to link object code files to create executable file

sudoku: sudoku.o Verifier.o

g++ $(CCFLAGS) -o sudoku sudoku.o Verifier.o

# Rules to compile source code files to object code

sudoku.o: sudoku.cpp Verifier.h

g++ $(CCFLAGS) -c sudoku.cpp

Verifier.o: Verifier.cpp Verifier.h

g++ $(CCFLAGS) -c Verifier.cpp

# Pseudo-target to remove object code and executable files

clean:

-rm \*.o sudoku

**Extra Credit (10 points)**

Code your printGrid() method to print the Sudoku grid array in a slightly fancier form, similar to the following:

-------------------------

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

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

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

-------------------------

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

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

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

-------------------------

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

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

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

-------------------------

**Sample output**



THANK YOU!