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

README

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

James DiLorenzo

Grace Meilen

Israel Hilerio

Lab 4

Sudoku:

We ran into trouble using the debug window to read and write to files. We ended up doing all of our debugging with strategically placed cout lines.

We chose to store our file as follows: The first line is the game name, and the second line is a string of characters representing the pieces. Zeroes and blank spaces are read as blanks, while all other special characters are ignored. When loading sudoku0, we first concatenated all of the lines in the file to a single string, deleted all spaces, and changed all zeroes to ‘ ‘ before parsing the string.

Because we read one character at a time and ignore everything else, strange characters should get ignored. Excess characters should get ignored because we only read the first 81 characters in the second line. We will start a new game if the first line in the file doesn’t match the game name. This is as intended, as it prevents users from accidentally loading poorly labeled files in their directory.

**Copy Constructor**

We use a shallow copy constructor for all three of the classes. This should be fine because each execution of lab4.exe can only play one game at a time. That is, because we should never have more than one Sudoku game being played simultaneously, we should never have issues with writing to the wrong existing game.

**Destructor**

We let C++ handle the destructor for all three of the classes. There is nothing clever to be done with the destructor as when the game is over, the exe file finishes its execution.

**Move Constructer.**

For the move constructor, we explicitly forbid it in each of the three classes. For example, in the Gomoku header, we had the following code in our class declaration:

Gomoku(Gomoku &&g) = delete;

This is acceptable because there should never be any use for the move constructor, it is simply not necessary to implement. There should never be a case when you have one game created and want to transfer it to another object. You should only ever be playing one game at a time.

**Copy Assignment Operator**

For all three of the classes, we allowed the implicitly defined copy assignment operator to remain instead of defining our own. This version of the copy assignment operator, for every member, performs copy assignments of the object’s bases and non-static members. This is sufficient for our purposes.

**Move Assignment Operator**

Much life for the move constructor, we explicitly forbid them in each of the header classes for the 3 game types. There should never be any reason to use a move assignment operator or, at least, we never ran into a need for one.