Introduction:

Sudoku is a popular puzzle where a player completes a grid (see below) based on some (but not all) cells provided. The challenge is to find the values of the unknown cells.

Figure 1: A Sudoku puzzle (From Will Shortz, Sudoku: Easy to Hard, St. Martin’s Griffin, 2005).

Rules:

A traditional Sudoku puzzle is either a 4x4, 9x9 or 16x16 grid. In the case of a 9x9 numeric puzzle, a solved puzzle has the following three properties:

• Every column contains the symbols 1–9 exactly once
• Every row contains the symbols 1–9 exactly once
• Each of the 3x3 minigrids contains the symbols 1–9 exactly once.
Over the next two weeks, we will implement simple but generic Sudoku code (this lab) and develop a solving algorithm with the help of the STL (Lab 6).

**Input format for lab:**

Puzzles will be composed of 9 lines, each of which contains 9 values separated by a space. For purposes of our software, values are either 1–9 (filled) or 0 (empty).

A sample puzzle based on Figure 1 (previous page) is encoded below:

```
0 3 2 0 0 8 9 1 4
0 0 0 0 0 0 0 3
0 0 7 1 0 0 0 2 6
0 0 8 0 7 6 0 0 0
9 2 1 3 0 0 0 8 7
0 6 0 0 0 4 0 0
0 0 0 0 0 0 0
0 0 0 4 0 5 0 8
8 0 0 6 0 7 0 0
```

**Checkpoint:** Using BitBucket / Github to share code

1. Sign up for Github or Bitbucket if you have not already.

2. If you are not in Basic Unix for Engineers this term, or took it prior to 2016, follow the tutorial on your choice, and ask TAs if you have questions. We will try and provide some input prior to the lab period but feel as if the Linux-based tutorials below are well done and definitely the place to start:
   - Git: https://help.github.com/articles/set-up-git/
   - Bitbucket: https://www.atlassian.com/git/tutorials/setting-up-a-repository

3. Show an active repository to a TA before leaving lab. This can be from a prior assignment/project, or simply a brief description of this lab assignment uploaded to the repository if you are new to these tools.

4. We expect many of you will have plenty of time due to synergy with Basic to Unix, so please continue to the Sudoku game portion in lab once checking off.

**Part 1:** A general framework for Sudoku

Many alternatives to Sudoku exist. Develop a templated Puzzle class such that you can easily create a Puzzle<int> (traditional Sudoku) or a Puzzle<char> (for an alternative Wordoku; see Coder challenge)

Provide a simple driver (main.cpp) that reads in the provided Sudoku and Wordoku
puzzle files using a Puzzle constructor, stores it in a vector of vectors (2D array) that is a private data member of the Puzzle class, and simply displays the unfinished puzzles to the screen. You can assume the “Sudoku” puzzle is just ints, and the “Wordoku” puzzle is char for this example.

**Part 2: Making a simple game**

Develop an interactive game that will read in a traditional (1-9; 0 = empty) Sudoku puzzle and ask a user to enter in a specific cell to replace its value. Your simple game must do the following:

- Use your templated class from Part 1
- Determine if a given entry is valid based on the current state of the board, i.e., that number does not occur in the same row, column or mini grid
- Determine if a puzzle is completed successfully

We will only test your simple game with a text file derived from Figure 1 (see sample).

In your usual report, add a paragraph on what you think worked well, what did not work, and a general summary of the “how” it was implemented. This is effectively more detail than usual on how the programmer works, but from the programmer perspective.

**Coder challenge!**

Wordoku is a variant of Sudoku where the numbers 1 .. n for a n x n puzzle are replaced with n specific letters (see http://en.wikipedia.org/wiki/Sudoku)

Modify/develop your simple game for Wordoku puzzles, where the first n entries of the input file are the values for the puzzle (numbers or letters) and the templated class is able to both check valid moves and determine if the puzzle is correct. You will receive up to $10 coder dollars each for both functions for arbitrary Wordoku puzzles of size 9 X 9.