# Sudoku-3: Game

CSCI 4526 / 6626 Fall 2019

## 1 Goals

- To create a simple text-based interface for Sudoku Helper
- To open and read part of a file, and to pass the rest as a parameter.
- To provide an interactive way to test Board and the other classes.

## 2 The Game Class

Game will be the controller class for this application. Your unit test should not be code this time, it should be plan (a text file) of what interactive inputs to give to the Game and what the results should be.

First, main() program must get a file name from the command line and pass it as a parameter to the Game constructor. Instantiate Game from main, then call Game::run(). This code will be part of your final project.

**Game Data Members.** At this time, the data members will be:

- A menu, which is a const array of quoted strings. Syntax should be something like this:
  `const char* menu[6]={"MenuItem1", "MenuItem 2", ...`
  It makes a program easier to modify if things like this are defined as separate structures at the top of a module, not embedded in the code.
- A Board*. You have not yet defined the Board class, but you can define a pointer to a Board by using a forward declaration before the beginning of the Game class: class Board;
- Game size, a small integer. When the project is finished, 6 and 9 will be the only legal values for $N$.
- Game type, a character: When the project is finished, the legal type codes will be 't' for traditional, 'd' for diagonal, and 's' for sixy.
- An ifstream for the input file.

More data members will be added as the weeks go by.

**Define a Game constructor:**

- One parameter, a file name.
- Open the ifstream to read the named file. (Call `fatal()` if this step fails.)
- Read and store a single letter from the first line of the file. This is the type code. Do not use getline. You now have all the information necessary to allocate a board, but the Board class will not be created until next week. At that time, you will add code to read the rest of the file.

  To validate an alphabetic menu selection, Define a string of all the legal menu selection codes. For example, if an application had three actions (go, stop, quit) then the string would be `"GSQ"`. To make this handle both cases, use a string like this: `"GgSsQq"`. Use a function from the c++ string library to search this string for the type code. If it is found, the type is legal. If not, there is a fatal error in the file.

- Define `Game::run()`. In this function, write a loop that will display the menu (details given below) and ask the user what to do next. For now, code a switch statement that does nothing except when Quit is selected. End the menu loop when the user selects *Quit*. Code will be added next week to handle the actions that result from choosing "Mark".

### 2.1 The Menu

We will implement two menu entries this week, more later. Use the *menu_c()* function in tools to implement the menu. To use this function, you must supply a title, the number of menu choices, an array of const char* for the options, and a const char* c-string that lists the first letter of each legal choice. This string will be used by *menu_c* to validate user selections. The menu should display these options; only the first and last will be implemented now.

- Mark: Input row, column, and value, then call Board::mark().
- Undo
- Redo
- Save game
- Restore game
- Quit and discard game: End and print a message.

## 3 Due September 17

Submit a screen shot of the menu as well as the usual parts.