70083 ExerciseTypes.CW1

C++ 1

Submitters

sf23 Shihan Fu

76/100

Good job!

```
Sudoku: Summary for sf23 of v5
------------------------------

  Comparison with Model Answer:
    Task 1:              1 / 1
    Task 2:              0 / 1
    Task 3:              1 / 1
    Task 4:              1 / 1
    Task 5:              1 / 1
    Additional puzzles:  1 / 1

Git Repo: git@gitlab.doc.ic.ac.uk:lab2324_autumn/msc_lab1_sf23.git
Commit ID: 77667
```

```
Detailed Output for test: Task 1
--------------------------------

Task 1

  Compiled OK

  Compilation Standard Output:

g++ -Wall -g main.cpp sudoku.cpp -o sudoku

  Test Passed




Detailed Output for test: Task 2
--------------------------------

Task 2

  Compiled OK

  Compilation Standard Output:

g++ -Wall -g main.cpp sudoku.cpp -o sudoku

  Test failed because Output differs

  Model Output (Left) vs Student’s Output (Right):

=================== Question 2 ===================                                          =================== Question 2 ===================

Loading Sudoku board from file ’easy.dat’... Success!                                       Loading Sudoku board from file ’easy.dat’... Success!
Putting ’1’ into I8 is a valid move. The board is:                                          Putting ’1’ into I8 is a valid move. The board is:
    1   2   3   4   5   6   7   8   9                                                           1   2   3   4   5   6   7   8   9
  +===========+===========+===========+                                                       +===========+===========+===========+
A |   :   :   | 1 :   : 8 | 3 :   :   |                                                     A |   :   :   | 1 :   : 8 | 3 :   :   |
  +---+---+---+---+---+---+---+---+---+                                                       +---+---+---+---+---+---+---+---+---+
B | 2 : 4 :   |   : 5 :   |   :   :   |                                                     B | 2 : 4 :   |   : 5 :   |   :   :   |
  +---+---+---+---+---+---+---+---+---+                                                       +---+---+---+---+---+---+---+---+---+
C |   :   : 8 |   :   :   |   : 6 : 1 |                                                     C |   :   : 8 |   :   :   |   : 6 : 1 |
  +===========+===========+===========+                                                       +===========+===========+===========+
D |   :   : 4 |   :   : 9 |   :   : 3 |                                                     D |   :   : 4 |   :   : 9 |   :   : 3 |
  +---+---+---+---+---+---+---+---+---+                                                       +---+---+---+---+---+---+---+---+---+
E |   : 6 :   |   :   :   |   : 2 :   |                                                     E |   : 6 :   |   :   :   |   : 2 :   |
  +---+---+---+---+---+---+---+---+---+                                                       +---+---+---+---+---+---+---+---+---+
F | 3 :   :   | 8 :   :   | 1 :   :   |                                                     F | 3 :   :   | 8 :   :   | 1 :   :   |
  +===========+===========+===========+                                                       +===========+===========+===========+
G | 1 : 7 :   |   :   :   | 9 :   :   |                                                     G | 1 : 7 :   |   :   :   | 9 :   :   |
  +---+---+---+---+---+---+---+---+---+                                                       +---+---+---+---+---+---+---+---+---+
H |   :   :   |   : 1 :   |   : 5 : 2 |                                                     H |   :   :   |   : 1 :   |   : 5 : 2 |
  +---+---+---+---+---+---+---+---+---+                                                       +---+---+---+---+---+---+---+---+---+
I |   :   : 2 | 7 :   : 4 |   : 1 :   |                                                     I |   :   : 2 | 7 :   : 4 |   : 1 :   |
  +===========+===========+===========+                                                       +===========+===========+===========+
Loading Sudoku board from file ’easy.dat’... Success!                                       Loading Sudoku board from file ’easy.dat’... Success!

Putting ’3’ into F8 is NOT a valid move. The board is:                                      Putting ’3’ into F8 is NOT a valid move. The board is:
    1   2   3   4   5   6   7   8   9                                                           1   2   3   4   5   6   7   8   9
  +===========+===========+===========+                                                       +===========+===========+===========+
A |   :   :   | 1 :   : 8 | 3 :   :   |                                                     A |   :   :   | 1 :   : 8 | 3 :   :   |
  +---+---+---+---+---+---+---+---+---+
B | 2 : 4 :   |   : 5 :   |   :   :   |
  +---+---+---+---+---+---+---+---+---+
C |   :   : 8 |   :   :   |   : 6 : 1 |
  +===========+===========+===========+
D |   :   : 4 |   :   : 9 |   :   : 3 |
  +---+---+---+---+---+---+---+---+---+
E |   : 6 :   |   :   :   |   : 2 :   |
  +---+---+---+---+---+---+---+---+---+
F | 3 :   :   | 8 :   :   | 1 :   :   |
  +===========+===========+===========+
G | 1 : 7 :   |   :   :   | 9 :   :   |
  +---+---+---+---+---+---+---+---+---+
H |   :   :   |   : 1 :   |   : 5 : 2 |
  +---+---+---+---+---+---+---+---+---+
I |   :   : 2 | 7 :   : 4 |   :   :   |
  +===========+===========+===========+
Loading Sudoku board from file ’easy.dat’... Success!

Putting ’3’ into B5 is NOT a valid move. The board is:
    1   2   3   4   5   6   7   8   9
  +===========+===========+===========+
A |   :   :   | 1 :   : 8 | 3 :   :   |
  +---+---+---+---+---+---+---+---+---+
B | 2 : 4 :   |   : 5 :   |   :   :   |
  +---+---+---+---+---+---+---+---+---+
C |   :   : 8 |   :   :   |   : 6 : 1 |
  +===========+===========+===========+
D |   :   : 4 |   :   : 9 |   :   : 3 |
  +---+---+---+---+---+---+---+---+---+
E |   : 6 :   |   :   :   |   : 2 :   |
  +---+---+---+---+---+---+---+---+---+
F | 3 :   :   | 8 :   :   | 1 :   :   |
  +===========+===========+===========+
G | 1 : 7 :   |   :   :   | 9 :   :   |
  +---+---+---+---+---+---+---+---+---+
H |   :   :   |   : 1 :   |   : 5 : 2 |
  +---+---+---+---+---+---+---+---+---+
I |   :   : 2 | 7 :   : 4 |   :   :   |
  +===========+===========+===========+
Loading Sudoku board from file ’easy.dat’... Success!

Putting ’0’ into E5 is NOT a valid move. The board is:
    1   2   3   4   5   6   7   8   9
  +===========+===========+===========+
A |   :   :   | 1 :   : 8 | 3 :   :   |
  +---+---+---+---+---+---+---+---+---+
B | 2 : 4 :   |   : 5 :   |   :   :   |
  +---+---+---+---+---+---+---+---+---+
C |   :   : 8 |   :   :   |   : 6 : 1 |
  +===========+===========+===========+
D |   :   : 4 |   :   : 9 |   :   : 3 |
  +---+---+---+---+---+---+---+---+---+
E |   : 6 :   |   :   :   |   : 2 :   |
  +---+---+---+---+---+---+---+---+---+
F | 3 :   :   | 8 :   :   | 1 :   :   |
  +===========+===========+===========+
G | 1 : 7 :   |   :   :   | 9 :   :   |
  +---+---+---+---+---+---+---+---+---+
H |   :   :   |   : 1 :   |   : 5 : 2 |
  +---+---+---+---+---+---+---+---+---+
```

```
    +---+---+---+---+---+---+---+---+---+
  B | 2 : 4 :   |   : 5 :   |   :   :   |
    +---+---+---+---+---+---+---+---+---+
  C |   :   : 8 |   :   :   |   : 6 : 1 |
    +===========+===========+===========+
  D |   :   : 4 |   :   : 9 |   :   : 3 |
    +---+---+---+---+---+---+---+---+---+
  E |   : 6 :   |   :   :   |   : 2 :   |
    +---+---+---+---+---+---+---+---+---+
  F | 3 :   :   | 8 :   :   | 1 :   :   |
    +===========+===========+===========+
  G | 1 : 7 :   |   :   :   | 9 :   :   |
    +---+---+---+---+---+---+---+---+---+
  H |   :   :   |   : 1 :   |   : 5 : 2 |
    +---+---+---+---+---+---+---+---+---+
  I |   :   : 2 | 7 :   : 4 |   :   :   |
    +===========+===========+===========+
  Loading Sudoku board from file ’easy.dat’... Success!

| Putting ’3’ into B5 is a valid move. The board is:
      1   2   3   4   5   6   7   8   9
    +===========+===========+===========+
  A |   :   :   | 1 :   : 8 | 3 :   :   |
    +---+---+---+---+---+---+---+---+---+
| B | 2 : 4 :   |   : 3 :   |   :   :   |
    +---+---+---+---+---+---+---+---+---+
  C |   :   : 8 |   :   :   |   : 6 : 1 |
    +===========+===========+===========+
  D |   :   : 4 |   :   : 9 |   :   : 3 |
    +---+---+---+---+---+---+---+---+---+
  E |   : 6 :   |   :   :   |   : 2 :   |
    +---+---+---+---+---+---+---+---+---+
  F | 3 :   :   | 8 :   :   | 1 :   :   |
    +===========+===========+===========+
  G | 1 : 7 :   |   :   :   | 9 :   :   |
    +---+---+---+---+---+---+---+---+---+
  H |   :   :   |   : 1 :   |   : 5 : 2 |
    +---+---+---+---+---+---+---+---+---+
  I |   :   : 2 | 7 :   : 4 |   :   :   |
    +===========+===========+===========+
  Loading Sudoku board from file ’easy.dat’... Success!

  Putting ’0’ into E5 is NOT a valid move. The board is:
      1   2   3   4   5   6   7   8   9
    +===========+===========+===========+
  A |   :   :   | 1 :   : 8 | 3 :   :   |
    +---+---+---+---+---+---+---+---+---+
  B | 2 : 4 :   |   : 5 :   |   :   :   |
    +---+---+---+---+---+---+---+---+---+
  C |   :   : 8 |   :   :   |   : 6 : 1 |
    +===========+===========+===========+
  D |   :   : 4 |   :   : 9 |   :   : 3 |
    +---+---+---+---+---+---+---+---+---+
  E |   : 6 :   |   :   :   |   : 2 :   |
    +---+---+---+---+---+---+---+---+---+
  F | 3 :   :   | 8 :   :   | 1 :   :   |
    +===========+===========+===========+
  G | 1 : 7 :   |   :   :   | 9 :   :   |
    +---+---+---+---+---+---+---+---+---+
  H |   :   :   |   : 1 :   |   : 5 : 2 |
    +---+---+---+---+---+---+---+---+---+
I |   :   : 2 | 7 :   : 4 |   :   :   |
  +===========+===========+===========+
Loading Sudoku board from file 'easy.dat'... Success!

Putting 'Z' into E5 is NOT a valid move. The board is:
    1   2   3   4   5   6   7   8   9
  +===========+===========+===========+
A |   :   :   | 1 :   : 8 | 3 :   :   |
  +---+---+---+---+---+---+---+---+---+
B | 2 : 4 :   |   : 5 :   |   :   :   |
  +---+---+---+---+---+---+---+---+---+
C |   :   : 8 |   :   :   |   : 6 : 1 |
  +===========+===========+===========+
D |   :   : 4 |   :   : 9 |   :   : 3 |
  +---+---+---+---+---+---+---+---+---+
E |   : 6 :   |   :   :   |   : 2 :   |
  +---+---+---+---+---+---+---+---+---+
F | 3 :   :   | 8 :   :   | 1 :   :   |
  +===========+===========+===========+
G | 1 : 7 :   |   :   :   | 9 :   :   |
  +---+---+---+---+---+---+---+---+---+
H |   :   :   |   : 1 :   |   : 5 : 2 |
  +---+---+---+---+---+---+---+---+---+
I |   :   : 2 | 7 :   : 4 |   :   :   |
  +===========+===========+===========+
Loading Sudoku board from file 'easy.dat'... Success!
Putting '8' into II is NOT a valid move. The board is:
    1   2   3   4   5   6   7   8   9
  +===========+===========+===========+
A |   :   :   | 1 :   : 8 | 3 :   :   |
  +---+---+---+---+---+---+---+---+---+
B | 2 : 4 :   |   : 5 :   |   :   :   |
  +---+---+---+---+---+---+---+---+---+
C |   :   : 8 |   :   :   |   : 6 : 1 |
  +===========+===========+===========+
D |   :   : 4 |   :   : 9 |   :   : 3 |
  +---+---+---+---+---+---+---+---+---+
E |   : 6 :   |   :   :   |   : 2 :   |
  +---+---+---+---+---+---+---+---+---+
F | 3 :   :   | 8 :   :   | 1 :   :   |
  +===========+===========+===========+
G | 1 : 7 :   |   :   :   | 9 :   :   |
  +---+---+---+---+---+---+---+---+---+
H |   :   :   |   : 1 :   |   : 5 : 2 |
  +---+---+---+---+---+---+---+---+---+
I |   :   : 2 | 7 :   : 4 |   :   :   |
  +===========+===========+===========+
Loading Sudoku board from file 'easy.dat'... Success!
Putting '5' into B0 is NOT a valid move. The board is:
    1   2   3   4   5   6   7   8   9
  +===========+===========+===========+
A |   :   :   | 1 :   : 8 | 3 :   :   |
  +---+---+---+---+---+---+---+---+---+
B | 2 : 4 :   |   : 5 :   |   :   :   |
  +---+---+---+---+---+---+---+---+---+
C |   :   : 8 |   :   :   |   : 6 : 1 |
  +===========+===========+===========+
D |   :   : 4 |   :   : 9 |   :   : 3 |
  +---+---+---+---+---+---+---+---+---+
E |   : 6 :   |   :   :   |   : 2 :   |
  +---+---+---+---+---+---+---+---+---+
```

```
I |   :   : 2 | 7 :   : 4 |   :   :   |
  +===========+===========+===========+
Loading Sudoku board from file 'easy.dat'... Success!

Putting 'Z' into E5 is NOT a valid move. The board is:
    1   2   3   4   5   6   7   8   9
  +===========+===========+===========+
A |   :   :   | 1 :   : 8 | 3 :   :   |
  +---+---+---+---+---+---+---+---+---+
B | 2 : 4 :   |   : 5 :   |   :   :   |
  +---+---+---+---+---+---+---+---+---+
C |   :   : 8 |   :   :   |   : 6 : 1 |
  +===========+===========+===========+
D |   :   : 4 |   :   : 9 |   :   : 3 |
  +---+---+---+---+---+---+---+---+---+
E |   : 6 :   |   :   :   |   : 2 :   |
  +---+---+---+---+---+---+---+---+---+
F | 3 :   :   | 8 :   :   | 1 :   :   |
  +===========+===========+===========+
G | 1 : 7 :   |   :   :   | 9 :   :   |
  +---+---+---+---+---+---+---+---+---+
H |   :   :   |   : 1 :   |   : 5 : 2 |
  +---+---+---+---+---+---+---+---+---+
I |   :   : 2 | 7 :   : 4 |   :   :   |
  +===========+===========+===========+
Loading Sudoku board from file 'easy.dat'... Success!
Putting '8' into II is NOT a valid move. The board is:
    1   2   3   4   5   6   7   8   9
  +===========+===========+===========+
A |   :   :   | 1 :   : 8 | 3 :   :   |
  +---+---+---+---+---+---+---+---+---+
B | 2 : 4 :   |   : 5 :   |   :   :   |
  +---+---+---+---+---+---+---+---+---+
C |   :   : 8 |   :   :   |   : 6 : 1 |
  +===========+===========+===========+
D |   :   : 4 |   :   : 9 |   :   : 3 |
  +---+---+---+---+---+---+---+---+---+
E |   : 6 :   |   :   :   |   : 2 :   |
  +---+---+---+---+---+---+---+---+---+
F | 3 :   :   | 8 :   :   | 1 :   :   |
  +===========+===========+===========+
G | 1 : 7 :   |   :   :   | 9 :   :   |
  +---+---+---+---+---+---+---+---+---+
H |   :   :   |   : 1 :   |   : 5 : 2 |
  +---+---+---+---+---+---+---+---+---+
I |   :   : 2 | 7 :   : 4 |   :   :   |
  +===========+===========+===========+
Loading Sudoku board from file 'easy.dat'... Success!
Putting '5' into B0 is NOT a valid move. The board is:
    1   2   3   4   5   6   7   8   9
  +===========+===========+===========+
A |   :   :   | 1 :   : 8 | 3 :   :   |
  +---+---+---+---+---+---+---+---+---+
B | 2 : 4 :   |   : 5 :   |   :   :   |
  +---+---+---+---+---+---+---+---+---+
C |   :   : 8 |   :   :   |   : 6 : 1 |
  +===========+===========+===========+
D |   :   : 4 |   :   : 9 |   :   : 3 |
  +---+---+---+---+---+---+---+---+---+
E |   : 6 :   |   :   :   |   : 2 :   |
  +---+---+---+---+---+---+---+---+---+
F | 3 :   :   | 8 :   :   | 1 :   :   |
  +===========+===========+===========+
G | 1 : 7 :   |   :   :   | 9 :   :   |
  +---+---+---+---+---+---+---+---+---+
H |   :   :   |   : 1 :   |   : 5 : 2 |
  +---+---+---+---+---+---+---+---+---+
I |   :   : 2 | 7 :   : 4 |   :   :   |
  +===========+===========+===========+
```

```
F | 3 :   :   | 8 :   :   | 1 :   :   |
  +===========+===========+===========+
G | 1 : 7 :   |   :   :   | 9 :   :   |
  +---+---+---+---+---+---+---+---+---+
H |   :   :   |   : 1 :   |   : 5 : 2 |
  +---+---+---+---+---+---+---+---+---+
I |   :   : 2 | 7 :   : 4 |   :   :   |
  +===========+===========+===========+
```

```


Detailed Output for test: Task 3
--------------------------------

Task 3

  Compiled OK

  Compilation Standard Output:

g++ -Wall -g main.cpp sudoku.cpp -o sudoku

  Test Passed




Detailed Output for test: Task 4
--------------------------------

Task 4

  Compiled OK

  Compilation Standard Output:

g++ -Wall -g main.cpp sudoku.cpp -o sudoku

  Test Passed




Detailed Output for test: Task 5
--------------------------------

Task 5

  Compiled OK

  Compilation Standard Output:

g++ -Wall -g main.cpp sudoku.cpp -o sudoku

  Test Passed




Detailed Output for test: Additional puzzles
--------------------------------------------

```

```
Additional puzzles

  Compiled OK

  Compilation Standard Output:

g++ -Wall -g main.cpp sudoku.cpp -o sudoku

  Test Passed


```

```cpp
#ifndef SUDOKU_H
#define SUDOKU_H
#include <cstring>
#include <string>

```

You should not change the function declarations we give you to "string". It is much better to learn how to use char array c-strings at the moment, as they will help with your understanding of the underlying data!

```cpp
using namespace std;

/* pre-supplied function to load a Sudoku board from a file */
void load_board(const char *filename, char board[9][9]);

/* pre-supplied function to display a Sudoku board */
void display_board(const char board[9][9]);

/*  checks whether all board positions are occupied by digits, and false
 * otherwise. No logical check*/
bool is_complete(const char board[9][9]);

/* check if it is valid to put digit in a given position */
bool make_move(string position, char digit, char board[9][9]);

/* check if it is valid to put digit in a given position with index and board
 * area check*/
bool make_move_index(int row, int col, char digit, char board[9][9]);

/* save the data in a file */
bool save_board(string filename, char board[9][9]);

/* solve the sudoku puzzle with recursions */
bool solve_board(char board[9][9]);

#endif
```

Nice to see some function comments, but more detail would be better!

```cpp
#include "sudoku.h"
#include <cassert>
#include <cstdio>
#include <cstring>
#include <fstream>
#include <iostream>

using namespace std;

/* You are pre-supplied with the functions below. Add your own
  function definitions to the end of this file. */

/* pre-supplied function to load a Sudoku board from a file */
void load_board(const char *filename, char board[9][9]) {

  cout << "Loading Sudoku board from file '" << filename << "'... ";

  ifstream in(filename);
  if (!in) {
    cout << "Failed!\n";
  }
  assert(in);

  char buffer[512];

  int row = 0;
  in.getline(buffer, 512);
  while (in && row < 9) {
    for (int n = 0; n < 9; n++) {
      assert(buffer[n] == '.' || isdigit(buffer[n]));
      board[row][n] = buffer[n];
    }
    row++;
    in.getline(buffer, 512);
  }

  cout << ((row == 9) ? "Success!" : "Failed!") << '\n';
  assert(row == 9);
}

/* internal helper function */
void print_frame(int row) {
  if (!(row % 3)) {
    cout << "  +===========+===========+===========+\n";
  } else {
    cout << "  +---+---+---+---+---+---+---+---+---+\n";
  }
}

/* internal helper function */
void print_row(const char *data, int row) {
  cout << (char)('A' + row) << " ";
  for (int i = 0; i < 9; i++) {
    cout << ((i % 3) ? ':' : '|') << " ";
    cout << ((data[i] == '.') ? ' ' : data[i]) << " ";
  }
  cout << "|\n";
}

/* pre-supplied function to display a Sudoku board */
void display_board(const char board[9][9]) {
  cout << "    ";
  for (int r = 0; r < 9; r++) {
    cout << (char)('1' + r) << "   ";
  }
  cout << '\n';
  for (int r = 0; r < 9; r++) {
    print_frame(r);
    print_row(board[r], r);
  }
  print_frame(9);
}

/* my own functions */

/* check if it is valid to put digit in a given position */
bool make_move(string position, char digit, char board[9][9]) {
  // check input position's length
  if (position.length() > 3) {
    return false;
  }
  // check position's content
  if (position[0] >= 'A' && position[0] <= 'I' && position[1] >= '1' &&
      position[1] <= '9') {
    // the position is good
```

This should be restructured to not have an empty if block. This is not very elegant.

```
  } else {
    return false;
  }

  // check digit
  if (digit < '1' || digit > '9') {
```

17/25 Pretty good

```
    return false;
  }

  // check logic
  int row = position[0] - 65;
```

Please use ASCII literals in times like this!

```
  int col = position[1] - 48 - 1;
  for (int i = 0; i < 9; i++) {

    if (i != col && board[row][i] == digit) {
      // row check fail
      return false;
```

-4 Did not check if digit appears in the current 3x3 block

```
    }

    if (i != row && board[i][col] == digit) {
      // column check fail
      return false;
    }
  }

  // update the board
  board[row][col] = digit;
```

-4 Did not check if position is already filled

```
  return true;
}

/*  checks whether all board positions are occupied by digits, and false
 * otherwise. No logical check*/
bool is_complete(const char board[9][9]) {
  for (int i = 0; i < 9; i++) {
    for (int j = 0; j < 9; j++) {
      // if is blank, will be replaced by a '.'
      if (board[i][j] == '.') {
        return false;
```

You don't actually need this first check.

```
      }
      // if is not a digit between 1 to 9
      if (board[i][j] < '1' || board[i][j] > '9') {
        return false;
      }
    }
```

10/10 Looks good

```
  }
  return true;
}
```

```

/* save the data in a file */
bool save_board(string filename, char board[9][9]) {
  ofstream output;
  output.open(filename);
  if (!output) {
    // fali to open file
```

Good to check if file opened correctly...

```
    return false;
  }
  for (int i = 0; i < 9; i++) {
    for (int j = 0; j < 9; j++) {
      // put chars into the output stream
```

12/15 Good job

```
      output.put(board[i][j]);
    }
    output.put('\n');
  }
  output.close();
```

-3 But also need to check if anything went wrong while writing!

```
  return true;
}

/* solve the sudoku puzzle with recursions */
bool solve_board(char board[9][9]) {
```

Putting all the logic inside the nested loops here does work, given your return statements in the if statement, but it isn't particularly neat.

```
  for (int i = 0; i < 9; i++) {
    for (int j = 0; j < 9; j++) {
      // find am empty cell
      if (board[i][j] == '.') {
        // try to put in a number from 1 to 9
        for (char number = '1'; number <= '9'; number++) {
          if (make_move_index(i, j, number, board)) {
            // if valid operation
            board[i][j] = number;
            if (solve_board(board)) {
              // recur the board
              return true;
            }
            // if put in this number cannot lead to a valid solution, replace it
            board[i][j] = '.';
          }
        }
        // if no valid operation, trace back
```

24/30 This is very close to working, but the make_move_index logic is not quite correct.

```
        return false;
      }
    }
  }
  // if there is no empty cells, find a solution
  return true;
}

/* check if it is valid to put digit in a given position with index and board
 * area check*/
bool make_move_index(int row, int col, char digit, char board[9][9]) {
  for (int i = 0; i < 9; i++) {
    // if the digit appeared in the tor ow column or the block area, the move
    // cannot be successful
    if (board[row][i] == digit || board[i][col] == digit ||
        board[3 * (row / 3) + i / 3][3 * (col / 3) + i % 3] == digit) {
      return false;
    }
```

This logic is not actually checking the 3x3 block correctly - it is only checking the diagonal of the block.

```
  }
  return true;
}
```

```
#include <iostream>
#include <cstdio>
#include<cstring>
#include<vector>
#include <fstream>
#include <cassert>
#include <time.h>
#include "sudoku.h"

using namespace std;

int main() {

  char board[9][9];



  /* This section illustrates the use of the pre-supplied helper functions. */
  // cout << "============= Pre-supplied functions =============\n\n";

  // cout << "Calling load_board():\n";
  // load_board("easy.dat", board);

  // cout << '\n';
  // cout << "Displaying Sudoku board with display_board():\n";
  // display_board(board);
  // cout << "Done!\n\n";


  // cout << "=================== Question 1 ===================\n\n";

  // load_board("easy.dat", board);
  // cout << "Board is ";
  // if (!is_complete(board)) {
  //   cout << "NOT ";
  // }
  // cout << "complete.\n\n";

  // load_board("easy-solution.dat", board);
  // cout << "Board is ";
  // if (!is_complete(board)) {
  //   cout << "NOT ";
  // }
  // cout << "complete.\n\n";

  // cout << "=================== Question 2 ===================\n\n";

  // load_board("easy.dat", board);

  // // test 1
  // cout << "Putting '1' into I8 is ";

  // if (!make_move("I8", '1', board)) {
  //   cout << "NOT ";
  // }
  // cout << "a valid move. The board is:\n";
  // display_board(board);

  // //write more tests

  // // test 2
  // cout << "Putting '0' into I8 is ";

  // if (!make_move("I8", '0', board)) {
  //   cout << "NOT ";
  // }
```

```
  // cout << "a valid move. The board is:\n";
  // display_board(board);

  // // test 3
  // cout << "Putting '9' into Z5 is ";

  // if (!make_move("Z5", '9', board)) {
  //   cout << "NOT ";
  // }
  // cout << "a valid move. The board is:\n";
  // display_board(board);





  // cout << "=================== Question 3 ===================\n\n";

  // load_board("easy.dat", board);
  // if (save_board("easy-copy.dat", board)) {
  //   cout << "Save board to 'easy-copy.dat' successful.\n";
  // } else {
  //   cout << "Save board failed.\n";
  // }
  // cout << '\n';

  // cout << "=================== Question 4 ===================\n\n";


  // load_board("easy.dat", board);

  // if (solve_board(board)) {
  //   cout << "The 'easy' board has a solution:\n";
  //   display_board(board);
  // } else {
  //   cout << "A solution cannot be found.\n";
  // }
  // cout << '\n';

  // load_board("medium.dat", board);
  // if(solve_board(board)){
  //   cout << "The 'medium' board has a solution:\n";
  //   display_board(board);
  // } else {
  //   cout << "A solution cannot be found.\n";
  // }
  // cout << '\n';

  // write more tests
  // load_board("mystery2.dat", board);

  // if(solve_board(board)){
  //   cout << "The 'mystery2' board has a solution:\n";
  //   display_board(board);
  // } else {
  //   cout << "A solution cannot be found.\n";
  //   cout << "This is the origin board.\n";
  //   load_board("mystery2.dat", board);
  //   display_board(board);

  // }

  // cout << '\n';

  // cout << "=================== Question 5 ===================\n\n";

  // // write more tests
  clock_t tStart = clock();

  load_board("mystery2.dat", board);

  for(int i=0;i<100;i++){
    solve_board(board);
  }

  clock_t tEnd = clock();
  if(solve_board(board)){
    cout << "The 'mystery2' board has a solution:\n";
    display_board(board);
  } else {
    cout << "A solution cannot be found.\n";
  }

  cout << '\n';

  cout << "The time used is : " << tEnd - tStart << endl;



  return 0;
}
```

```
sudoku: main.cpp sudoku.cpp sudoku.h
	g++ -Wall -g main.cpp sudoku.cpp -o sudoku
```

```
Summarise your findings here (see specification).
Q5  Before the validation, I define how to judge whether a puzzle is "hard". We think the more recursions one puzzle needs, the more difficult it is.
    If a puzzle needs more recursions to find a solution, it needs more time to execute. So I used the "time.h" to help me calculate the running time.
    I set two variables to record the time when the function of solve_board is started and it ended and substracting them to get the time duration of executing.
    In order to minimize the random CPU error, I used for loops to execute the function for multiple times.(100 in this case)

    program result:
    mystery1.dat  ->SUCESS 92995
    mystery2.dat  ->CANNOT BE FOUND
    mystery3.dat  ->SUCESS 3172

    conclusion: according to the time used to solve these puzzles,
        mystery1 is extremely hard
        mystery2 is impossible to solve
        mystery3 is hard
```

You have correctly identified the boards, but I would like to see some more analysis here!

Why is your method of averaging the times a good measure of difficulty? What are you actually measuring? Is it the efficiency of your solve_board function?

What makes one board more difficult to solve than another? Does the way your algorithm is written affect that?

Also, whilst averaging the timings is a reasonable method, I would have liked to have seen some recursion call counting.

13/20