

70083 ExerciseTypes.CW1

C++ 1

Submitters

sf23

Shihan Fu

76/100

Good job!

Emarking

Sudoku**TestSummary.txt: 1/1****Shihan Fu - sf23:v5**

```
1: Sudoku: Summary for sf23 of v5
2: -----
3:
4:   Comparison with Model Answer:
5:   Task 1:           1 / 1
6:   Task 2:           0 / 1
7:   Task 3:           1 / 1
8:   Task 4:           1 / 1
9:   Task 5:           1 / 1
10:  Additional puzzles: 1 / 1
11:
12: Git Repo: git@gitlab.doc.ic.ac.uk:lab2324_autumn/msc_lab1_sf23.git
13: Commit ID: 77667
```

Sudoku

results_iconv.txt: 1/5

Shihan Fu - sf23:v5

```
1: Detailed Output for test: Task 1
2: -----
3:
4: Task 1
5:
6:   Compiled OK
7:
8:   Compilation Standard Output:
9:
10: g++ -Wall -g main.cpp sudoku.cpp -o sudoku
11:
12:   Test Passed
13:
14:
15:
16:
17: Detailed Output for test: Task 2
18: -----
19:
20: Task 2
21:
22:   Compiled OK
23:
24:   Compilation Standard Output:
25:
26: g++ -Wall -g main.cpp sudoku.cpp -o sudoku
27:
28:   Test failed because Output differs
29:
30:   Model Output (Left) vs Student's Output (Right):
31:
32: ===== Question 2 =====
33:
34: Loading Sudoku board from file 'easy.dat'... Success!
35: Putting '1' into I8 is a valid move. The board is:
36:   1  2  3  4  5  6  7  8  9
37: +-----+-----+-----+
38: A |   :   :   | 1 :   : 8 | 3 :   :   |
39: +-----+-----+-----+
40: B | 2 : 4 :   |   : 5 :   |   :   :   |
41: +-----+-----+-----+
42: C |   :   : 8 |   :   :   |   : 6 : 1 |
43: +-----+-----+-----+
44: D |   :   : 4 |   :   : 9 |   :   : 3 |
45: +-----+-----+-----+
46: E |   : 6 :   |   :   :   |   : 2 :   |
47: +-----+-----+-----+
48: F | 3 :   :   | 8 :   :   | 1 :   :   |
49: +-----+-----+-----+
50: G | 1 : 7 :   |   :   :   | 9 :   :   |
51: +-----+-----+-----+
52: H |   :   :   |   : 1 :   |   : 5 : 2 |
53: +-----+-----+-----+
54: I |   :   : 2 | 7 :   : 4 |   : 1 :   |
55: +-----+-----+-----+
56: Loading Sudoku board from file 'easy.dat'... Success!
57:
58: Putting '3' into F8 is NOT a valid move. The board is:
59:   1  2  3  4  5  6  7  8  9
60: +-----+-----+-----+
61: A |   :   :   | 1 :   : 8 | 3 :   :   |
```

```
===== Question 2 =====

Loading Sudoku board from file 'easy.dat'... Success!
Putting '1' into I8 is 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 |   : 1 :   |
+-----+-----+-----+
Loading Sudoku board from file 'easy.dat'... Success!

Putting '3' into F8 is NOT a valid move. The board is:
  1  2  3  4  5  6  7  8  9
+-----+-----+-----+
A |   :   :   | 1 :   : 8 | 3 :   :   |
```

Sudoku

results_iconv.txt: 2/5

Shihan Fu - sf23:v5

```
62: +-----+-----+-----+
63: B | 2 : 4 : | : 5 : | : : |
64: +-----+-----+-----+
65: C | : : 8 | : : | : 6 : 1 |
66: +-----+-----+-----+
67: D | : : 4 | : : 9 | : : 3 |
68: +-----+-----+-----+
69: E | : 6 : | : : | : 2 : |
70: +-----+-----+-----+
71: F | 3 : : | 8 : : | 1 : : |
72: +-----+-----+-----+
73: G | 1 : 7 : | : : | 9 : : |
74: +-----+-----+-----+
75: H | : : | : 1 : | : 5 : 2 |
76: +-----+-----+-----+
77: I | : : 2 | 7 : : 4 | : : |
78: +-----+-----+-----+
79: Loading Sudoku board from file 'easy.dat'... Success!
80:
81: Putting '3' into B5 is NOT a valid move. The board is:
82:   1  2  3  4  5  6  7  8  9
83: +-----+-----+-----+
84: A | : : | 1 : : 8 | 3 : : |
85: +-----+-----+-----+
86: B | 2 : 4 : | : 5 : | : : |
87: +-----+-----+-----+
88: C | : : 8 | : : | : 6 : 1 |
89: +-----+-----+-----+
90: D | : : 4 | : : 9 | : : 3 |
91: +-----+-----+-----+
92: E | : 6 : | : : | : 2 : |
93: +-----+-----+-----+
94: F | 3 : : | 8 : : | 1 : : |
95: +-----+-----+-----+
96: G | 1 : 7 : | : : | 9 : : |
97: +-----+-----+-----+
98: H | : : | : 1 : | : 5 : 2 |
99: +-----+-----+-----+
100: I | : : 2 | 7 : : 4 | : : |
101: +-----+-----+-----+
102: Loading Sudoku board from file 'easy.dat'... Success!
103:
104: Putting '0' into E5 is NOT a valid move. The board is:
105:   1  2  3  4  5  6  7  8  9
106: +-----+-----+-----+
107: A | : : | 1 : : 8 | 3 : : |
108: +-----+-----+-----+
109: B | 2 : 4 : | : 5 : | : : |
110: +-----+-----+-----+
111: C | : : 8 | : : | : 6 : 1 |
112: +-----+-----+-----+
113: D | : : 4 | : : 9 | : : 3 |
114: +-----+-----+-----+
115: E | : 6 : | : : | : 2 : |
116: +-----+-----+-----+
117: F | 3 : : | 8 : : | 1 : : |
118: +-----+-----+-----+
119: G | 1 : 7 : | : : | 9 : : |
120: +-----+-----+-----+
121: H | : : | : 1 : | : 5 : 2 |
122: +-----+-----+-----+
```

```
+-----+-----+-----+
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 |
+-----+-----+-----+
```

Sudoku

results_iconv.txt: 3/5

Shihan Fu - sf23:v5

```
123: I | : : 2 | 7 : : 4 | : : |
124: +-----+-----+-----+
125: Loading Sudoku board from file 'easy.dat'... Success!
126:
127: Putting 'Z' into E5 is NOT a valid move. The board is:
128:   1  2  3  4  5  6  7  8  9
129: +-----+-----+-----+
130: A | : : : | 1 : : 8 | 3 : : |
131: +-----+-----+-----+
132: B | 2 : 4 : | : 5 : | : : : |
133: +-----+-----+-----+
134: C | : : 8 | : : : | : 6 : 1 |
135: +-----+-----+-----+
136: D | : : 4 | : : 9 | : : 3 |
137: +-----+-----+-----+
138: E | : 6 : | : : : | : 2 : |
139: +-----+-----+-----+
140: F | 3 : : | 8 : : | 1 : : |
141: +-----+-----+-----+
142: G | 1 : 7 : | : : : | 9 : : |
143: +-----+-----+-----+
144: H | : : : | : 1 : | : 5 : 2 |
145: +-----+-----+-----+
146: I | : : 2 | 7 : : 4 | : : : |
147: +-----+-----+-----+
148: Loading Sudoku board from file 'easy.dat'... Success!
149: Putting '8' into II is NOT a valid move. The board is:
150:   1  2  3  4  5  6  7  8  9
151: +-----+-----+-----+
152: A | : : : | 1 : : 8 | 3 : : |
153: +-----+-----+-----+
154: B | 2 : 4 : | : 5 : | : : : |
155: +-----+-----+-----+
156: C | : : 8 | : : : | : 6 : 1 |
157: +-----+-----+-----+
158: D | : : 4 | : : 9 | : : 3 |
159: +-----+-----+-----+
160: E | : 6 : | : : : | : 2 : |
161: +-----+-----+-----+
162: F | 3 : : | 8 : : | 1 : : |
163: +-----+-----+-----+
164: G | 1 : 7 : | : : : | 9 : : |
165: +-----+-----+-----+
166: H | : : : | : 1 : | : 5 : 2 |
167: +-----+-----+-----+
168: I | : : 2 | 7 : : 4 | : : : |
169: +-----+-----+-----+
170: Loading Sudoku board from file 'easy.dat'... Success!
171: Putting '5' into B0 is NOT a valid move. The board is:
172:   1  2  3  4  5  6  7  8  9
173: +-----+-----+-----+
174: A | : : : | 1 : : 8 | 3 : : |
175: +-----+-----+-----+
176: B | 2 : 4 : | : 5 : | : : : |
177: +-----+-----+-----+
178: C | : : 8 | : : : | : 6 : 1 |
179: +-----+-----+-----+
180: D | : : 4 | : : 9 | : : 3 |
181: +-----+-----+-----+
182: E | : 6 : | : : : | : 2 : |
183: +-----+-----+-----+
```

```
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 : |
+-----+-----+-----+
```

Sudoku

```
184: F | 3 : : | 8 : : | 1 : : |
185: +-----+-----+-----+
186: G | 1 : 7 : | : : | 9 : : |
187: +-----+-----+-----+
188: H | : : | : 1 : | : 5 : 2 |
189: +-----+-----+-----+
190: I | : : 2 | 7 : : 4 | : : |
191: +-----+-----+-----+
192:
193:
194: Detailed Output for test: Task 3
195: -----
196:
197: Task 3
198:
199:     Compiled OK
200:
201:     Compilation Standard Output:
202:
203: g++ -Wall -g main.cpp sudoku.cpp -o sudoku
204:
205:     Test Passed
206:
207:
208:
209:
210: Detailed Output for test: Task 4
211: -----
212:
213: Task 4
214:
215:     Compiled OK
216:
217:     Compilation Standard Output:
218:
219: g++ -Wall -g main.cpp sudoku.cpp -o sudoku
220:
221:     Test Passed
222:
223:
224:
225:
226: Detailed Output for test: Task 5
227: -----
228:
229: Task 5
230:
231:     Compiled OK
232:
233:     Compilation Standard Output:
234:
235: g++ -Wall -g main.cpp sudoku.cpp -o sudoku
236:
237:     Test Passed
238:
239:
240:
241:
242: Detailed Output for test: Additional puzzles
243: -----
244:
```

results_iconv.txt: 4/5

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

Shihan Fu - sf23:v5

Sudoku

results_iconv.txt: 5/5

Shihan Fu - sf23:v5

```
245: Additional puzzles
246:
247:   Compiled OK
248:
249:   Compilation Standard Output:
250:
251: g++ -Wall -g main.cpp sudoku.cpp -o sudoku
252:
253:   Test Passed
254:
255:
```

```

1: #ifndef SUDOKU_H
2: #define SUDOKU_H
3: #include <cstring>
4: #include <string>
5:
6: using namespace std;
7:
8: /* pre-supplied function to load a Sudoku board from a file */
9: void load_board(const char *filename, char board[9][9]);
10:
11: /* pre-supplied function to display a Sudoku board */
12: void display_board(const char board[9][9]);
13:
14: /* checks whether all board positions are occupied by digits, and false
15:  * otherwise. No logical check*/
16: bool is_complete(const char board[9][9]);
17:
18: /* check if it is valid to put digit in a given position */
19: bool make_move(string position, char digit, char board[9][9]);
20:
21: /* check if it is valid to put digit in a given position with index and board
22:  * area check*/
23: bool make_move_index(int row, int col, char digit, char board[9][9]);
24:
25: /* save the data in a file */
26: bool save_board(string filename, char board[9][9]);
27:
28: /* solve the sudoku puzzle with recursions */
29: bool solve_board(char board[9][9]);
30:
31: #endif

```

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

```

1: #include "sudoku.h"
2: #include <cassert>
3: #include <cstdio>
4: #include <cstring>
5: #include <fstream>
6: #include <iostream>
7:
8: using namespace std;
9:
10: /* You are pre-supplied with the functions below. Add your own
11:  * function definitions to the end of this file. */
12:
13: /* pre-supplied function to load a Sudoku board from a file */
14: void load_board(const char *filename, char board[9][9]) {
15:
16:     cout << "Loading Sudoku board from file '" << filename << "'... ";
17:
18:     ifstream in(filename);
19:     if (!in) {
20:         cout << "Failed!\n";
21:     }
22:     assert(in);
23:
24:     char buffer[512];
25:
26:     int row = 0;
27:     in.getline(buffer, 512);
28:     while (in && row < 9) {
29:         for (int n = 0; n < 9; n++) {
30:             assert(buffer[n] == '.' || isdigit(buffer[n]));
31:             board[row][n] = buffer[n];
32:         }
33:         row++;
34:         in.getline(buffer, 512);
35:     }
36:
37:     cout << ((row == 9) ? "Success!" : "Failed!") << '\n';
38:     assert(row == 9);
39: }
40:
41: /* internal helper function */
42: void print_frame(int row) {
43:     if (!(row % 3)) {
44:         cout << " +-----+-----+-----+\n";
45:     }
46:     cout << " +---+---+---+---+---+---+---+---+\n";
47: }
48: }
49:
50: /* internal helper function */
51: void print_row(const char *data, int row) {
52:     cout << (char)('A' + row) << " ";
53:     for (int i = 0; i < 9; i++) {
54:         cout << ((i % 3) ? ':' : '|') << " ";
55:         cout << ((data[i] == '.') ? ' ' : data[i]) << " ";
56:     }
57:     cout << "\n";
58: }
59:
60: /* pre-supplied function to display a Sudoku board */
61: void display_board(const char board[9][9]) {
62:     cout << " ";
63:     for (int r = 0; r < 9; r++) {
64:         cout << (char)('1' + r) << " ";
65:     }
66:     cout << '\n';

```


Sudoku

sudoku.cpp: 2/3

Shihan Fu - sf23:v5

```

67: for (int r = 0; r < 9; r++) {
68:     print_frame(r);
69:     print_row(board[r], r);
70: }
71: print_frame(9);
72: }
73:
74: /* my own functions */
75:
76: /* check if it is valid to put digit in a given position */
77: bool make_move(string position, char digit, char board[9][9]) {
78:     // check input position's length
79:     if (position.length() > 3) {
80:         return false;
81:     }
82:     // check position's content
83:     if (position[0] >= 'A' && position[0] <= 'I' && position[1] >= '1' &&
84:         position[1] <= '9') {
85:         // the position is good
86:     } else {
87:         return false;
88:     }
89:
90:     // check digit
91:     if (digit < '1' || digit > '9') {
92:         return false;
93:     }
94:
95:     // check logic
96:     int row = position[0] - 65;
97:     int col = position[1] - 48 - 1;
98:     for (int i = 0; i < 9; i++) {
99:
100:         if (i != col && board[row][i] == digit) {
101:             // row check fail
102:             return false;
103:         }
104:
105:         if (i != row && board[i][col] == digit) {
106:             // column check fail
107:             return false;
108:         }
109:     }
110:
111:     // update the board
112:     board[row][col] = digit;
113:     return true;
114: }
115:
116: /* checks whether all board positions are occupied by digits, and false
117:  * otherwise. No logical check*/
118: bool is_complete(const char board[9][9]) {
119:     for (int i = 0; i < 9; i++) {
120:         for (int j = 0; j < 9; j++) {
121:             // if is blank, will be replaced by a '.'
122:             if (board[i][j] == '.') {
123:                 return false;
124:             }
125:             // if is not a digit between 1 to 9
126:             if (board[i][j] < '1' || board[i][j] > '9') {
127:                 return false;
128:             }
129:         }
130:     }
131:     return true;
132: }

```

17/25 Pretty good

Please use ASCII literals in times like this!

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

-4 Did not check if position is already filled

You don't actually need this first check.

10/10 Looks good

```
git@gitlab.doc.ic.ac.uk:lab2324_autumn/msc_lab1_sf23.git
```

77667

Sudoku

sudoku.cpp: 3/3

Shihan Fu - sf23:v5

```

133:
134: /* save the data in a file */
135: bool save_board(string filename, char board[9][9]) {
136:     ofstream output;
137:     output.open(filename);
138:     if (!output) {
139:         // fail to open file Good to check if file opened correctly...
140:         return false;
141:     }
142:     for (int i = 0; i < 9; i++) {
143:         for (int j = 0; j < 9; j++) {
144:             // put chars into the output stream 12/15 Good job
145:             output.put(board[i][j]);
146:         }
147:         output.put('\n');
148:     }
149:     output.close();
150:     return true; -3 But also need to check if anything went wrong while writing!
151: }
152:
153: /* solve the sudoku puzzle with recursions */
154: bool solve_board(char board[9][9]) {
155:     for (int i = 0; i < 9; i++) {
156:         for (int j = 0; j < 9; j++) {
157:             // find an empty cell
158:             if (board[i][j] == '.') {
159:                 // try to put in a number from 1 to 9
160:                 for (char number = '1'; number <= '9'; number++) {
161:                     if (make_move_index(i, j, number, board)) {
162:                         // if valid operation
163:                         board[i][j] = number;
164:                         if (solve_board(board)) {
165:                             // recur the board
166:                             return true;
167:                         }
168:                         // if put in this number cannot lead to a valid solution, replace it
169:                         board[i][j] = '.';
170:                     }
171:                 }
172:                 // if no valid operation, trace back
173:                 return false;
174:             }
175:         }
176:     }
177:     // if there is no empty cells, find a solution
178:     return true;
179: }
180:
181: /* check if it is valid to put digit in a given position with index and board
182: * area check*/
183: bool make_move_index(int row, int col, char digit, char board[9][9]) {
184:     for (int i = 0; i < 9; i++) {
185:         // if the digit appeared in the row or column or the block area, the move
186:         // cannot be successful
187:         if (board[row][i] == digit || board[i][col] == digit ||
188:             board[3 * (row / 3) + i / 3][3 * (col / 3) + i % 3] == digit) {
189:             return false;
190:         }
191:     }
192:     return true;
193: }

```

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

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

```
git@gitlab.doc.ic.ac.uk:lab2324_autumn/msc_lab1_sf23.git
```

77667

```

1: #include <iostream>
2: #include <cstdio>
3: #include<cstring>
4: #include<vector>
5: #include <fstream>
6: #include <cassert>
7: #include <time.h>
8: #include "sudoku.h"
9:
10: using namespace std;
11:
12: int main() {
13:
14:     char board[9][9];
15:
16:
17:
18:     /* This section illustrates the use of the pre-supplied helper functions. */
19:     // cout << "===== Pre-supplied functions =====\n\n";
20:
21:     // cout << "Calling load_board():\n";
22:     // load_board("easy.dat", board);
23:
24:     // cout << '\n';
25:     // cout << "Displaying Sudoku board with display_board():\n";
26:     // display_board(board);
27:     // cout << "Done!\n\n";
28:
29:
30:     // cout << "===== Question 1 =====\n\n";
31:
32:     // load_board("easy.dat", board);
33:     // cout << "Board is ";
34:     // if (!is_complete(board)) {
35:     //     cout << "NOT ";
36:     // }
37:     // cout << "complete.\n\n";
38:
39:     // load_board("easy-solution.dat", board);
40:     // cout << "Board is ";
41:     // if (!is_complete(board)) {
42:     //     cout << "NOT ";
43:     // }
44:     // cout << "complete.\n\n";
45:
46:     // cout << "===== Question 2 =====\n\n";
47:
48:     // load_board("easy.dat", board);
49:
50:     // // test 1
51:     // cout << "Putting '1' into I8 is ";
52:
53:     // if (!make_move("I8", '1', board)) {
54:     //     cout << "NOT ";
55:     // }
56:     // cout << "a valid move. The board is:\n";
57:     // display_board(board);
58:
59:     // //write more tests
60:
61:     // // test 2
62:     // cout << "Putting '0' into I8 is ";
63:
64:     // if (!make_move("I8", '0', board)) {
65:     //     cout << "NOT ";
66:     // }

```

```

67: // cout << "a valid move. The board is:\n";
68: // display_board(board);
69:
70: // // test 3
71: // cout << "Putting '9' into Z5 is ";
72:
73: // if (!make_move("Z5", '9', board)) {
74: //     cout << "NOT ";
75: // }
76: // cout << "a valid move. The board is:\n";
77: // display_board(board);
78:
79:
80:
81:
82:
83: // cout << "===== Question 3 =====\n\n";
84:
85: // load_board("easy.dat", board);
86: // if (save_board("easy-copy.dat", board)) {
87: //     cout << "Save board to 'easy-copy.dat' successful.\n";
88: // } else {
89: //     cout << "Save board failed.\n";
90: // }
91: // cout << '\n';
92:
93: // cout << "===== Question 4 =====\n\n";
94:
95:
96: // load_board("easy.dat", board);
97:
98: // if (solve_board(board)) {
99: //     cout << "The 'easy' board has a solution:\n";
100: //     display_board(board);
101: // } else {
102: //     cout << "A solution cannot be found.\n";
103: // }
104: // cout << '\n';
105:
106: // load_board("medium.dat", board);
107: // if (solve_board(board)) {
108: //     cout << "The 'medium' board has a solution:\n";
109: //     display_board(board);
110: // } else {
111: //     cout << "A solution cannot be found.\n";
112: // }
113: // cout << '\n';
114:
115: // write more tests
116: // load_board("mystery2.dat", board);
117:
118: // if (solve_board(board)) {
119: //     cout << "The 'mystery2' board has a solution:\n";
120: //     display_board(board);
121: // } else {
122: //     cout << "A solution cannot be found.\n";
123: //     cout << "This is the origin board.\n";
124: //     load_board("mystery2.dat", board);
125: //     display_board(board);
126:
127: // }
128:
129: // cout << '\n';
130:
131: // cout << "===== Question 5 =====\n\n";
132:

```

Sudoku

main.cpp: 3/3

Shihan Fu - sf23:v5

```
133: // // write more tests
134: clock_t tStart = clock();
135:
136: load_board("mystery2.dat", board);
137:
138: for(int i=0;i<100;i++){
139:     solve_board(board);
140: }
141:
142: clock_t tEnd = clock();
143: if(solve_board(board)){
144:     cout << "The 'mystery2' board has a solution:\n";
145:     display_board(board);
146: } else {
147:     cout << "A solution cannot be found.\n";
148: }
149:
150: cout << '\n';
151:
152: cout << "The time used is : " << tEnd - tStart << endl;
153:
154:
155:
156: return 0;
157: }
158:
159:
160:
161:
162:
```

Sudoku

makefile: 1/1

Shihan Fu - sf23:v5

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

```
1: Summarise your findings here (see specification).
2: 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.
3: 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.
4: I set two variables to record the time when the function of solve_board is /
started and it ended and subtracting them to get the time duration of executing.
5: In order to minimize the random CPU error, I used for loops to execute the /
function for multiple times.(100 in this case)
6:
7: program result:
8: mystery1.dat ->SUCESS 92995
9: mystery2.dat ->CANNOT BE FOUND
10: mystery3.dat ->SUCESS 3172
11:
12: conclusion: according to the time used to solve these puzzles,
13: mystery1 is extremely hard
14: mystery2 is impossible to solve
15: 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