CMSC223 Systems Programming - Lab 4

Arrays Practice – Part 2 (2-Dimensional Arrays)
To be completed by Monday, October 7.

Write a program (in a file, `fifteen.c`) that allows a user to play the "Fifteen puzzle". For example, see: http://migo.sixbit.org/puzzles/fifteen/. Your program does *not* need to *solve* the puzzle; it just lets the user play.

Your puzzle will start out in this state:

```
 1 10 15  4
13  6  3  8
 2  9 12  7
14  5    11
```

Your program will print out the puzzle (as above), ask the user what tile they would like to move, and then move the tile, repeating until the puzzle is solved. When the puzzle is solved, it reports so, and quits.

Here is the start of an example session:

```
  1 10 15  4
 13  6  3  8
  2  9 12  7
 14  5    11

Enter the tile you would like to move: 12

  1 10 15  4
 13  6  3  8
  2  9     7
 14  5 12 11

Enter the tile you would like to move: 7

  1 10 15  4
 13  6  3  8
  2  9  7
 14  5 12 11

Enter the tile you would like to move: 4
I can't move that tile.

  1 10 15  4
 13  6  3  8
  2  9  7
 14  5 12 11
```

```
Enter the tile you would like to move: 18
Invalid tile.

  1 10 15  4
 13  6  3  8
  2  9  7
 14  5 12 11


Enter the tile you would like to move: 11

  1 10 15  4
 13  6  3  8
  2  9  7 11
 14  5 12

Enter the tile you would like to move:-1
```

Entering `-1` quits the program.

With the starting puzzle as above, the following sequence of moves will solve it:

```
12 9 2 13 6 3 15 10 3 2 5 14 13 5 9 15 10 4 8 7 11 12 15 10 7 8 4 3 2
6 5 9 10 11 12
```

You may paste that line into an empty file and then feed it to your program (using < on the command line) to check your algorithm for detecting when the puzzle is solved.

When printing your puzzle board, it should be aligned like you see above. `printf` formatting codes work well for this.

Do not try to write the entire program all at once. Before setting out to write the program, think about what its structure and components will look like. Develop an incremental strategy to begin and complete the implementation in steps. Here is a breakdown:

Write functions to do the following:

`show(…)`: prints the puzzle board (as shown above)
`getMove(…)`: inputs a move from user
`move(…)`: makes a given move on the board, if possible

Then stitch them together into a main program:

1. initialize a board
2. show the board
3. while there is a move entered
3.1 make the move on the board
3.2 show the updated board