Memory Game

For this exercise, you will create a simple version of the Memory Game. If you are not familiar with this game, you can try playing this on-line version of it:

http://www.web-games-online.com/memory/

You will again be working with multiple functions and arrays. You will be using pointers for your arrays and you **must** use a **struct** to store the move and pass it to functions as needed.

Program Design

You may want to create two different 4x4 arrays. One to store the symbols the player is trying to match and the second one to be used to show the current state of the game. You could do it with a single array for the symbols, as an alternative design.

For the symbols, you can use any eight different characters. Possible options are the digits 1-8, the letters A-H, or a collection of special characters such as: !,@,#,$,%,^,&,*. The only requirement is that there are eight of them.

To initialize the symbol array, you will randomly place two copies of each symbol in your 4x4 array.

For each turn, you display the current state of the board, showing an X or a SPACE at each location. Each location is an X at the start of the game, SPACES are used to indicate where a symbol pair was found.

After showing the current state, you ask for a move. Then you redisplay the state, but at the chosen location, you show the real symbol instead of an X. Then you ask for a second move and display the board with those two locations showing the real symbol instead of X's. If the two symbols showing are a match, you replace the X's with SPACES. If they are not a match, you do nothing.

Once all X's have been replaced by SPACES, the game is done.

Program Requirements

You need to have a function to initialize your memory game and return it to main. This can be done with dynamic allocation of the array (as a 1x16 element array instead of a 4x4 one). By using a 1x16 array, it makes it simpler to determine the location to place items.

You need a function to get a move from the player and return it to main. This should ask for a row and column (each between 0 and 3). Validate that each one is in range. Return a struct containing the move as two integers. You can return this as a pointer or by copy, your choice.

You will need a function to display the board. You will pass one or two moves in to this function depending on if it is the first time or the second time. If you want, you can have two separate functions so that you have one with one move and one with two moves. (Remember, you can have two functions with the same name as long as they have different parameter lists.)

Your program should allow the player to repeat playing if they want. You can keep track of how many moves it took if you would like.

Memory Game

Program Hints

You can either have a separate array that is composed of X's and then converted to SPACES as items are matched or you can have one array and your output program only output an X when the location being displayed is not a SPACE and is not the one that was chosen.

If you have one array, then you replace the symbols with SPACES. If you have two arrays, you modify the ones with X's in it by storing SPACES over the X's.

The only parts of this assignment that are new are using a struct to pass moves around and writing the display function. You might want to write it all in main and get it working, then create functions and move code to them. As usual, get your create array and display array functions working first. Then you can add move and update board.

Since you created a 1x16 array, but you are accessing it as a 4x4 array, you will need to use the formula index = row * ROW_LEN + col. For example, if they wanted to see the location 3, 0 (which is the bottom row, leftmost item), it would be at index = 3 * 4 + 0 or 12. If you count the locations starting at 0, you find this is correct.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

For more on this calculation, look at the Moodle document on flattening an array.