CS 162
Programming Lab 4

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

Program Description

While the program is an introduction to the use of pointers and dynamic memory, it is also a review of overloaded functions and multi-dimension arrays. The implementation described in here is not the only way to do such a program, but this approach is designed to provide practice in specific concepts

For this program there will a dynamic array of characters created and passed around. This array will be a flattened two-dimensional array.

For the symbols, use the characters A, B, C, D, E, F, G, H, I, J, K, L, M, N, O. They should be defined in an array that is in the function to create a board. The smaller board will use the characters A-H and the larger version will use all the characters.

For each turn, display the current state of the board, showing an UNKNOWN or a SPACE at each location. Each location starts being shown as UNKNOWN. Once the value at a location has been discovered, a SPACE will be shown there.

After showing the current state, get a move. Then redisplay the state, but at the move's location, show the correct symbol instead of UNKNOWN. Then get a second move and display the board with each of the two locations showing the correct symbol instead of UNKNOWN. A move is stored as a two-element array containing the row and column of the move.

Then check for a match; if there is a match update the board replacing the matched characters with a SPACE at the two locations that were compared.

The game will end after the player has cleared the board.

For this, as in all assignments, you are expected to follow the course programming style guidelines and to properly implement both header and source files for your functions. All input should be validated as integer and within range.

Program Requirements

This program should include at least the following functions and constants.

Main

The main program should have two loops, the outer loop repeats with a new game as long as the player wants to keep playing, the inner loop continues until the game ends with 8 matches.

The main program needs variables for move1, move2, board (a pointer), rowLength, colLength, and a counter of how many tries that is updated each time through the guessing loop.

To start the game, main will ask whether to have a hard game or an easy game. For a hard game, the board will be 30 squares (5 x 6) and for an easy game it will be 16 (4 x 4).

Recommended constants

HARD_ROW = 5, HARD_COL = 6, EASY_ROW = 4, EASY_COL = 4, SPACE = ‘ ‘, UNKNOWN = ‘X’

Helper Functions

- getInteger
  - used to get an input value
  - input parameters are min and max value
  - get and validates an integer between min and max
  - return the value

- yesOrNo
  - used to respond to a question
  - input parameter is a string message to display
  - get input, verify if yes or no (y/n)
  - return true for yes, false for no

- getIndex
  - inline function that is used to simplify access to the flattened board
  - input parameters are a move (two element integer array) and colLength
  - converts the row and column of the move into a 1D index
  - returns the index 0-15

- getIndex (overloaded version)
  - inline function that is used to simplify access to the flattened board
  - input parameters are a rowIndex, colIndex, and colLength
  - converts the row and column of the move into a 1D index
  - returns the index 0-15
  -

Game Functions

- displayInstructions
  - no input parameters
  - displays the instructions for the game
    - detailed the first time, simplified all later times
  - no return values

- createBoard
  - pass in a variable, 1 for easy, 2 for hard
  - dynamically creates a board of the proper size of characters
  - places two of each character into the board
  - calls shuffle, passing the board, to randomize the locations
  - returns the board

- shuffle
  - input parameter is a board (passed by pointer) and total length
  - randomly shuffles the board, see suggestion below
  - no return values, board is updated by reference

- getMove
    - input is the board, rowLength, colLength, and the two-element array for move1
    - get two integers (row, col), validate that they are in range (0-3)
    - verify that the location is not SPACE
    - save the move in the parameter
    - no return values

- getMove (overloaded)
    - input is the board, rowLength, colLength, move1, and an array for move2
    - get two integers, validate that they are in range
    - verify the new move is to a different location than move1
    - verify that the location is not SPACE
    - update move2 parameter
    - no return values

- showBoard
    - input parameters are the board, rowLength, and colLength,
    - display the board as a 4x4 grid using nested for loops
        - for each cell in the board
            - if it is SPACE, show it
            - if not SPACE, show UNKNOWN
    - no return values

- showBoard (overloaded)
    - input parameters are the board, rowLength, colLength, and move1
    - display the board as a 4x4 grid using nested for loops
        - for each cell other than move1
            - same as regular showBoard
        - for move1 display that location of board
    - no return values

- showBoard (overloaded)
    - input parameters are the board, rowLength, colLength, move1, and move2
    - display the board as a 4x4 grid using nested for loops
        - for cells other than move1 and move2
            - same as regular showBoard
        - for move1 and move2 display that location of board
    - no return values

- checkMatch
    - input parameters are the board, move1, move2
    - if the board at move1 has the same value as the board at move2 return true
    - else return false

- updateBoard
    - input parameters are the board, move1, move2
    - update the locations at move1 and move2 by storing SPACE
    - no return values

Program Hints

To shuffle the array, simply go through it and swap each item with a random other location in the array. You can write your own as a simple for loop or you can use the one in algorithms.

http://www.cplusplus.com/reference/algorithm/shuffle/