Assignment 3: 100 + 10 bonus marks

Topics: 1 and 2D arrays, functions using call-by-value and call-by-reference parameters

Submission Instructions:

- Submit a single C file containing your program. To submit, upload your C file to the submission box for A3 on moodle. Name your file as lastnameFirstnameA3.c (For example, if Ritu is the first name and Chaturvedi is the last name, the file would be called chaturvediRituA3.c). Incorrect file name will result in 10% penalty.
- Incorrect format of submitted files will result in automatic zero. (Must be a valid .c file)
- The program you submit must compile with no warnings and run successfully for full marks. You get a zero if your program doesn't compile. There is also a penalty for warnings (5% for each unique warning).
- Penalties will occur for missing style, comments, header comments etc.
    - Header comment must include your full name, username, c file name, last modified date in your header comments.
- DO NOT use global variables. Use of any global variables will result in automatic zero.
- DO NOT use goto statements. Use of any goto statements will result in automatic zero.
- You must follow the given function prototypes – any deviation will result in a penalty.
- You must use the main function as given –penalty will occur if any changes are made.
- You may use the zybooks style of implementing functions OR the style we use in our class and labs (using prototypes). Be consistent.

smartBoardA3

Tic-Tac-Toe is a board game that is very popular with kids. Typically, this game is played using a 3 X 3 board. Each player chooses a symbol – usually an 'X' or an 'O' and tries to be the first one to place 3 of his / her symbol in a straight line. This straight line could be horizontal (straight across a row), vertical (straight across a column), diagonal from left corner to the right corner or diagonal from right corner to the left. You may refer to Wikipedia to learn more about the game.

In this assignment, given the main and one other function definition (see smartBoardA3.c), you are asked to write several other functions that make up this game. Prototypes and description of these functions is given in the following pages. Note that an empty cell is represented by a '?' on the board. The game is played between a player and the computer and the player always goes first. It is important to understand the details of each function - this will guide you in coding the others.

Important tips:

1. You may want to start by writing code for functions createInitialBoard and printCurrentBoard. These are easiest to implement. Function printCurrentBoard will help to troubleshoot the rest of the assignment.

2. Read and understand the code (main and a function definition) given in smartBoardA3.c – they will help you get started on the assignment.
3. Start the assignment early – you already know most of the concepts required for this assignment.

Function names, their prototypes and description:

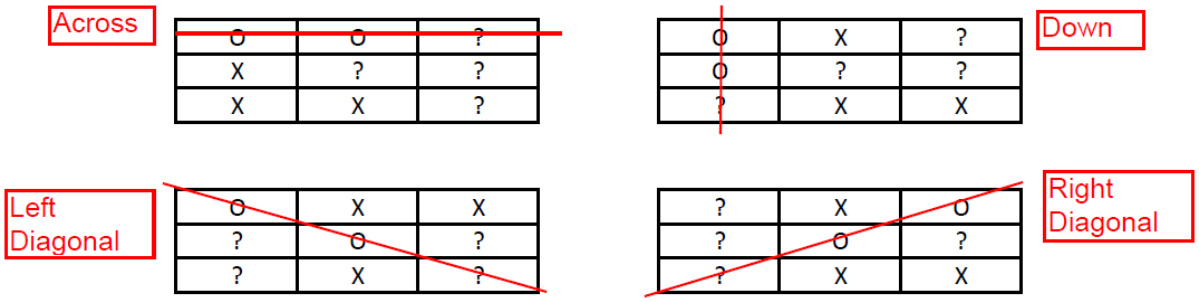
1	int isInputValid (int entered, int minimum, int maximum) Returns 1 if entered is between minimum and maximum; 0 otherwise
2	int isBoardFull (char board [n][n]); Returns 1 if there is no empty cell in the board; 0 otherwise
3	void createInitialBoard (char board [n][n]); Creates an empty board, which is a 2D array of size n X n – each cell on the board is assigned a '?'. It also prints the board. ? | ? | ? --- ? | ? | ? --- ? | ? | ?
4	void readUserMove (int * userRow, int * userCol) Prompts the user to input row and column values to place symbol X on the board
5	void printCurrentBoard (char board [n][n]); Prints the current board. You must use a nested loop to print the board.
6	int getComputerMove(char [n][n], int *, int *, int, int); generate a move for the computer - **Function definition given in smartBoardA3.c**
7	int gameWon (char board [n][n], char symbol); Returns 1 if there is a winner. This function is explained further on page on pages 3 and 4.
8	int computerPlaysToWin (char board [n][n], int * cRow, int * cCol); Returns 1 if the computer wins in this step. This function is explained further on pages 3 and 4.
9	void computerPlaysRandom (int * cRow, int * cCol, int uRow, int uCol); Sets the computer's move (position on the board for symbol 'O' in terms of row and column number). Parameters uRow and uCol are used to inform the computer of the

	user's move.
10	void sumAllDimensions (char board [n][n], int sumR[n], int sumC[n], int * sumLD, int * sumRD); This is a utility function that can be used by other functions. It computes the sum of scores across all rows, columns, left diagonal and right diagonal.
11	int memberOf (int aNum, int someArray [n]); Returns 1 if a value exists in the given array. For example, if someArray has the following values [10,20,30], then memberOf (30, someArray) returns 1 but memberOf (3, someArray) returns 0.

Details on some functions

1. It tries to win first (Function computerPlaysToWin)
   int computerPlaysToWin (char board[N][N], int * cRow, int * cCol);

In order to win, it checks whether the board has 2 'O' in any of the rows, columns or the diagonals. Some example scenarios below leads the computer to win:



One possible way of accomplishing this is discussed next:

a. assign a score of 4 to symbol 'O' on the board and 1 to symbol 'X'

b. If the total score for symbol 'O' in any of the rows or columns or diagonals is 8 (implying that 2 cells in that row / column / diagonal have symbol 'O'), then
   If the third cell in that row /column / diagonal is unoccupied (i.e. it has a '?'), then

   - the third unoccupied cell is assigned 'O'
   - computer wins and game ends

2. If the computer cannot win, then it plays a move inspired by the user's move (Function computerPlaysRandom)

void computerPlaysRandom (int * , int * , int , int);

- It tries to assign 'O' to a cell that is either on the same row or column or diagonal as the user's. It randomly picks a number between 0 and 2 – to decide whether it should use the same row as the user's current move OR the same column OR diagonal. If the cell it picks is occupied (by 'O' or 'X'), then it must repeat the process until it finds an empty cell (indicated by a '?').

3. Function gameWon:

int gameWon(char board[n][n], char symbol);

This function is called every time a user or a computer makes a move to check if there is a winner. The winning rule is similar to the one described in function computerPlaysToWin. Computer wins if the total score in any row / column / diagonal of the board is 12 (implying a count of 3 'O's in any single row or column or diagonal). Similarly, player wins if the total score in any row / column / diagonal of the board is 3 (implying a count of 3 'X's in any single row or column or diagonal).

BONUS Function (5 marks):

int computerPlaysToBlock (char board[N][N], int * cRow, int * cCol);

In order to block the user's move, it checks whether the board has 2 'X' in any of the rows, columns or the diagonals. Use a strategy similar to computerPlaysToWin. The function returns a 0 if it doesn't block.

So, If the total score for symbol 'X' in any of the rows or columns or diagonals is 2 (implying that 2 cells in that row / column / diagonal have symbol 'X'), then

If the third cell in that row /column / diagonal is unoccupied (i.e. it has a '?'), then the third unoccupied cell is assigned 'O'.

Function getComputerMove given to you has a commented part that you may use to test this function. You must uncomment that section if you wish to attempt the bonus function.

Sample Scenario:

Note that

- line numbers are given only for convenience – you DO NOT have to display them.
- This sample does not include the bonus function

```
1  Player's symbol: X
2  Computer's symbol: O
3
4  Here is the initial board - spaces are indicated by a ?
5  ? | ? | ?
6  ----------
7  ? | ? | ?
8  ----------
9  ? | ? | ?
10
11 Your move - enter numbers between 1 and 3
12
13 Enter row number: 2
14 Enter column number: 2
15
16 You chose row 2 and column 2
17
18 Computer chose row 1 and column 3
19
20 Current board now is:
21
22 ? | ? | O
23 ----------
24 ? | X | ?
25 ----------
26 ? | ? | ?
```

Continued on next page

```
27
28  Your move - enter numbers between 1 and 3
29
30  Enter row number: 1
31  Enter column number: 2
32
33  You chose row 1 and column 2
34
35  Computer chose row 1 and column 1
36
37  Current board now is:
38
39  O | X | O
40  ----------
41  ? | X | ?
42  ----------
43  ? | ? | ?
44
45  Your move - enter numbers between 1 and 3
46
47  Enter row number: 3
48  Enter column number: 2
49
50  You chose row 3 and column 2
51
52  Congrats - you won against the computer :) :
53
54  Current board now is:
55
56  O | X | O
57  ----------
58  ? | X | ?
59  ----------
60  ? | X | ?
```