EECS-22: Advanced C Programming (Winter 2025) Mid-term Exam, Part 2 (02/14/25)

Prepared by: Dr. Halima Bouzidi (EECS-22 Instructor)

Programming Problem

In this assignment, you will implement a simplified, modular Tic-Tac-Toe game in C. The requirements are:

- Use a 3×3 board (no larger).
- No pointers (e.g., no char * for board data) rely on fixed-size arrays.
- No file I/O; everything is input and output via standard input and output (scanf and printf).
- No getchar or similar functions; keep it basic.
- The user will play as two players, 'X' and 'O', taking turns.
- After each move, check if someone has won or if the board is full (tie).
- Print the board and the final result (win or tie).

You will organize your solution into three modules (each with a corresponding header file) plus a Makefile.

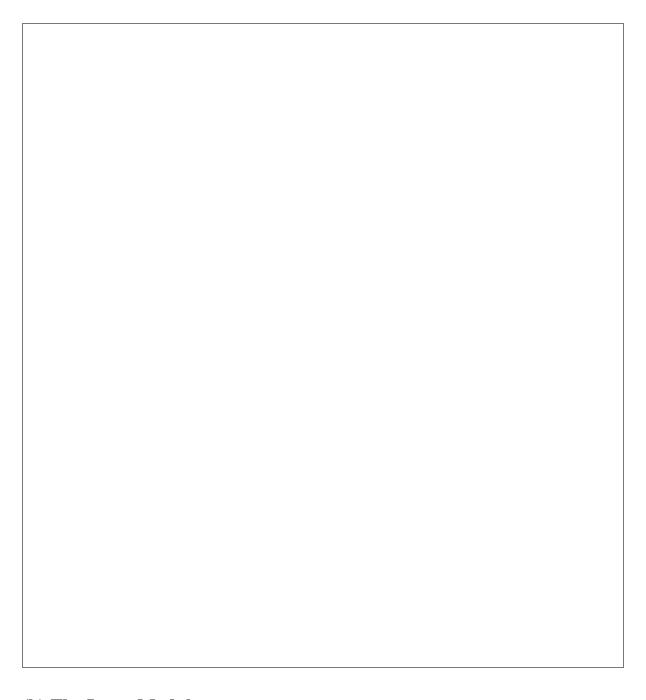
- 1. board.c and board.h (manages the tic-tac-toe board)
- 2. input.c and input.h (handles user input of row/column)
- 3. main.c (controls the game flow)
- 4. Makefile

(a) The Board Module

Create board.c with its header board.h to maintain the internal 3×3 grid:

- Use a static 2D array of chars (char board[3][3];).
- Initialize all cells to '.' (dot) to indicate an empty cell.
- Provide functions to set a cell (row, col) to ' $\tt X'$ or ' $\tt O'$.
- Provide a function to print the board to the screen.
- Provide a function CheckWinner that returns:
 - 'X' if X has won.
 - ' ○' if O has won,
 - 'T' if it's a tie (board is full, no winner),
 - '.' if the game is still ongoing (no winner yet, not full).

Required Functions (add more if needed): $/\star$ Initializes the board to all '.' (empty cells). $\star/$ void InitBoard(void); /* Sets the board[row][col] to symbol ('X' or 'O'), only if it's currently '.' (empty). $\star/$ void SetCell(int row, int col, char symbol); /* Prints the board (no pointers used). */ void PrintBoard(void); $/\star$ Checks if there's a winner or if it's a tie/ongoing. Returns 'X', 'O', 'T', or '.'. */ char CheckWinner(void); board.h (05 points) board.c (15 points)



(b) The Input Module

Implement input.c and input.h. You'll collect user moves (row and column) via scanf. Requirements:

- Read two integers (row and col) from the user.
- Return them to the caller for use in the game logic.
- No usage of getchar or pointer-based parameters.

Required Functions:

/* Reads row and col (0-based) from user,
 storing them in local variables only. Returns row, col through
 out-parameters or by separate calls. We do not use pointers.
 We might just do "return row" from one function, "return col"
 from another, or a single function returning an int that

```
encodes row/col somehow.
   (Here, we'll show a single function that returns row in
   readRow() and col in readCol().) */
int readRow(void);
int readCol(void);
input.h (05 points)
input.c (10 points)
```

(c) The main.c Program

Your main.c coordinates the game:

- 1. Initialize the board.
- 2. Print it.
- 3. Use a loop where players alternate between 'X' and 'O':
 - Read row, then column (0-based) using readRow() and readCol().
 - Set the chosen cell via SetCell (if it's empty).
 - Check for a winner with CheckWinner.
 - If winner or tie, break out and print the final result.
 - Otherwise, switch to the other player and continue.

main.c (15 points)					

(d) The Makefile Makefile (05 points)

Additional Hints & Tips

- **Keep it simple.** We only have a 3×3 grid, so no pointers or dynamic memory is needed.
- Zero-based indices. We read row 0 to 2 and col 0 to 2.
- Validate each move. If a player enters a row or column outside 0–2, print an error and ask again.
- Short game. This is just for practice, so there's no advanced UI or error recovery beyond basic checks.
- **Testing.** Check normal play, immediate wins (like row 0 filled), and tie scenarios.

Submission & Grading

- Submit all source files: main.c, board.c, board.h, input.c, input.h, and your Makefile.
- Your grade will focus on correctness, code clarity, and simple modular design without pointers or file I/O.