**Programming 1 – Fall 2020**  Student Name: \_\_\_\_\_\_\_Liam Martell\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**MP3 Scoresheet – 15 October 2020 (must be filled out)**

**Source Code Validation – to be turned in by END OF TODAY**

**Source Code (.cpp file) is to be submitted along with your scoresheet and screenshots.**

1. CODE CHECK - Adherence to specifications. You must have these functions with these parameters.

(Note: you can have more functions)

(5 point **DEDUCTION** for each violation):

|  |  |  |
| --- | --- | --- |
| **Function**  **Prototype?** | **Function used as specified?** | **NOTE: Function prototypes and**  **function use will be checked during grading** |
| **YES** | no | **YES** | no | **int getPlayerInput(string playerName)** |
| **YES** | no | **YES** | no | **bool isLegalMove(char board[SIZE][SIZE], int location)** |
| **YES** | no | **YES** | no | **void placeMarkOnBoard(char board[SIZE][SIZE], char playerMark, int location)** |
| **YES** | no | **YES** | no | **void clearBoard(char board[SIZE][SIZE])** |
| **YES** | no | **YES** | no | **bool hasThreeInRow(char board[SIZE][SIZE], char playerMark)** |
| **YES** | no | **YES** | no | **void displayBoard(char board[SIZE][SIZE])** |
| **YES** | no | **YES** | no | **void displayGameStats(int ties, int player1Score, int player2Score)**   * **may include strings for player names** |

Two-dimensional array used for board: **YES** | no

Global variables used (exempt: symbolic constants): yes | **NO**

**ALL STUDENTS ARE RESPONSIBLE TO DOUBLE-CHECK THAT THE REQUIRED FUNCTIONS ARE IMPLEMENTED AS STATED IN MP3.**

1. Demonstrate the following tests of your program via Screenshoets. Tests do not have to be in order and be combined within screenshots. Perform one complete run of the program, playing three games with your opponent. “Fix” the playing of your games such that X wins one game, O wins one game, and there is a tied game.

**Test #1: Player names requested and consistently used First Attempt Correct?** Y | N score: \_\_\_\_\_ / 5 points

**Test #2: Program detects win First Attempt Correct?** Y | N score: \_\_\_\_\_ / 5 points

**Test #3: Program alternates starting player First Attempt Correct?** Y | N score: \_\_\_\_\_ / 5 points

**Test #4: Program detects tied game First Attempt Correct?** Y | N score: \_\_\_\_\_ / 5 points

**Test #5: Program correctly keeps track of wins and ties First Attempt Correct?** Y | N score: \_\_\_\_\_ / 5 points

**Test #6: Program prevents selection of already played locations. First Attempt Correct?** Y | N score: \_\_\_\_\_ / 5 points

**Test #7: Program prevents selection of illegal positions. First Attempt Correct?** Y | N score: \_\_\_\_\_ / 5 points

**Test #8: Each player always plays the same mark (X or O) for all games. First Attempt Correct?** Y | N score: \_\_\_\_\_ / 5 points

***ALL FIRST ATTEMPTS CORRECT? YES | no*** score: \_\_\_\_\_ / 10 points

**Second Chance (to be initialed by instructor/Lab TA):**

Test #1: \_\_\_\_\_ Test #2: \_\_\_\_\_ Test #3: \_\_\_\_\_ Test #4: \_\_\_\_\_ Test #5: \_\_\_\_\_ Test #6: \_\_\_\_\_ Test #7: \_\_\_\_\_ Test #8: \_\_\_\_\_

**OTHER DEDUCTIONS:**

**Testing/Validation Subtotal: \_\_\_\_\_\_\_\_\_ / 50**

**Readability Subtotal: \_\_\_\_\_\_\_\_\_ / 30**

**Documentation Subtotal: \_\_\_\_\_\_\_\_\_ / 20**

**DEDUCTIONS: < \_\_\_\_\_\_\_\_\_\_ >**

**TOTAL: \_\_\_\_\_\_\_\_\_ / 100**

**Programming 1 – Fall 2020**  Student Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**MP3 Scoresheet – 15 October 2020 (must be filled out)**

**Source Code Analysis Rubric**

**Readability – 30 points total**

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Meets Expectations - 5** | **Needs Minor/Major Improvement – 4/3/2** | **Unacceptable/Missing - 0** |
| Organization  Score (x1): \_\_\_\_\_\_\_ | Code is broken down into clear, recognizable, well thought out sections of functional units; blank lines and comments used to establish visual structure. |  |  |
| Separation  Score (x1): \_\_\_\_\_\_\_ | Spaces used as appropriate to help differentiate distinct elements within each coding statement. |  |  |
| Alignment  Score (x1): \_\_\_\_\_\_\_ | Indentation emphasizes the body of an iterative or a conditional statement; braces and parentheses follow appropriate standards; start of comments are column-aligned as warranted. |  |  |
| Consistency  Score (x1): \_\_\_\_\_\_\_ | Similar coding constructs regularly use the same format regarding indentation and alignment; similar or related variable names follow an established pattern. |  |  |
| Nomenclature  Score (x2): \_\_\_\_\_\_\_ | All variables, save for common exceptions, have meaningful and informative names without being verbose; use of upper and lower case clearly differentiates variables, constants, and classes. |  |  |

**Documentation – 20 points total**

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Meets Expectations - 5** | **Needs Minor/Major Improvement – 4/3/2** | **Unacceptable/Missing - 0** |
| Program Header Comments  Score (x1): \_\_\_\_\_\_ | Every program starts with a header comment that contains the name of the file, the date of its writing, the full name of its author, and a description of what the program does. |  |  |
| Function Header Comments  Score (x1): \_\_\_\_\_\_\_ | All function header comments are with the definition (not prototype) and contain preconditions and postconditions as appropriate; each description clearly but succinctly explains purpose of the function. |  |  |
| Section Comments  Score (x1): \_\_\_\_\_\_\_ | Each functional section of code includes a comment describing the goal or purpose that that section is trying to accomplish without being either verbose or parroting. |  |  |
| Code Comments  Score (x1): \_\_\_\_\_\_\_ | Line-oriented comments are used to clarify meaning and/or provide elaboration as needed. |  |  |