Test Specification Dots and Boxes Program

1. Module Overview

The dots and boxes program allows a computer to play a game against a human opponent.

1.1 Assumptions/ pre-condition: User has an interest in playing the Dots and Boxes game and has the ability to execute the game.

1.2 Inputs

Data Names:
Size of matrix (integer variable in range)
Start position (combination one integer and one lowercase alphabet variable)
End position (combination one integer and one lowercase alphabet variable,
with one common value either integer or letter)
Values:
Size of matrix must be between two by two to eight by eight.
Start position must be a valid available "dot" on the grid created.
End position must be a valid available "dot" on the grid created, that must be within one row or column to "draw" a line.

1.2 Outputs

Ability to create working grid that matches user defined input.
Ability to quit game at any user turn.
Ability to create updated grid, drawing a connection line for the most current turn and all pervious turns, and updated score of both opponents.
Required to place a letter C for Computer or H for Human in a "box" completed by the correct opponent, containing four lines drawn to completion.
Lists game players' final scores and points earned, written to standard output.

2. Test Data

Representative Input

1. At least one typical input as described in the requirements document.

   a grid size , moves of two valid dots, or a quit

Functional Coverage

1. The player must be able to make grids sizes 2x2 to 8x8.
2. Valid user input must work.
3. Invalid user input must prompt a "sorry try again".
4. Invalid user input must ask again for the correct data.
5. Invalid user input is not allowed.
6. Valid user input must be accepted and stored into memory.
7. Grid rows must be labeled with letters.
8. Grid columns must be labeled with numbers.
9. Players must be able to make legal moves.
10. Player must be able to quit the game on their turn.
11. Game board must update each players move.
12. Game board must check for a full game board.
13. The game must be able to exit (in main.cpp return 0).
14. The game must recognize end of game condition.
15. Scores must display for both players correctly.
16. The beginning scores must be zero for both players.
17. Program must be able to determine available moves.
18. Program must be able to block unavailable moves.
19. Multiple attempts allowed to re-enter correct data on all fields.
20. The game must have the human player turn first.
21. The game must be able to switch players by assigning turns.
22. The program must accept keyboard user entry.
23. Program must be able to manage game logically (start, middle, end).
24. Program must be able to apply the move and update the score.
25. If the human player places the fourth edge a letter H must be displayed in the correct box.
26. The program must recognize game stages and progress to exit or to find winner.
27. The program must keep a running total of both players points.
28. The computer player must follow the 4 rules logic and in precedence.
29. Players must have the ability to make 1 or moves in each turn. (double moves)
30. Players must have ability to claim 2 boxes and display each box with the correct symbol.
31. The computer must try rule1, then rule 2, then rule 3, then rule 4 in order.
32. The program must be able to examine rule1 and decide execute code or exit condition to the next rule. (and repeat for all 4 rules)
33. The program must be able to count edges taken verses edges available.
34. The program must be able to create tempera arrays and sort as dictated by each of the 4 computer turn rules.
35. Program must allow double moves.
36. The program must be able to assign a winner or a draw scenario.
37. If the computer claims a box a letter C must be displayed in the correct box.
38. The computer must be the second player in the game.
39. The program must run correctly on the ODU UNIX system.

Boundary Values

1. The number of size smallest grid, test for size= 2 .
2. The number of largest grid size, test for size = 8 .
3. The number below smallest size, test for size=1 .
4. The number above largest size, test for size =9.
5. Check display grid is in bounds of user entry for size entered.
6. Check the H, C or blank space are displayed in the correct position.
7. Check that the edges are not out of bounds for the size of grid used.(off in space)
8. Verify edges are displayed at expected position.(a1 a2, is at a1 a2)
9. Check user input matches expected datatypes.
10. Verify arrays are not smaller than expected value.(too small)
11. Verify array size does not exceed expected value (too large)
12. Verify dot1 chosen is not outside of grid size.
13. Verify dot2 chosen is not outside of grid size.
14. Verify moves do not exceed number of moves predicted for grid size.(too many)
15. Verify moves do not fall short of predicted moves for grid size.(too few)

Special Values

1. Test input size of grid is size = 1.
2. Test input size of grid is size = - 1.(covers all negative integer issues with all programs)
3. Test input size of grid is size = 0.(covers division issues on all programs)
4. No infinite loops exist.
5. No complier errors exist.
6. Verify ability to read EOF, to read in data correctly from files.

(*Note: The special values are tests developers use in their toolbox that are known to be issues.)