**University of Pittsburgh**

**Super Connect Four**

**CS 1632**

**DELIVERABLE 1: Test Plan and Traceability Matrix**

Jamie Min – jam433  
Paul Yu – pyy2

Bill Laboon

08 Oct 2018

**Table of Contents**

1. INTRODUCTION …………………………………………………………… 3

2. TEST PLAN …………………………………………………………………. 4

3. TRACEABILITY MATRIX …………………………………………………10

4. DEFECTS …………………………………………………………………… 11

1. **Introduction**

**1.1 Test Overview**

*Super Connect Four* allows two players to play Connect Four against each other through the terminal with a twist. The board is represented as a square two-dimensional array with column numbers at the top. Given that the game is played through the terminal, most of the testing was regards to user inputs. Testing was broken down into 3 main categories: Program Execution, Game Play, and Exit Conditions.

**1.2 Program Execution**

Testing started with evaluating the program argument for validity. When determining board size, the argument should have only accepted non-negative integer values and the program should terminate with a message on all other inputs. Testing here included the edge argument 0, strings, and a large integer value. The large integer served as a corner case test due to the program being restricted to system memory. When creating a board size greater than 10, columns should be marked with the last integer i.e. ‘9’ for column ‘19’ instead of the entire integer value.

**1.3 Game Play**

When users start playing the game there are 2 types of moves to choose from. Players alternate to either place a marker using an integer value within bounds of the board size or they can “flip” or “rot.” The program should reject and display a prompt on any other inputs. Inputs were required to be case-insensitive which is how the defect “ROT” was discovered. Instead of rotating the board, the move dropped the corresponding player marker in the left most column. If the left column was full, the program threw a runtime error. Another defect was found while game play testing. When a floating-point value was used as a move the program re-prompted the player. Going back to program execution, when a floating-point argument was used in the command-line, the program proceeded to initialize. Extensively testing the player move allowed.

**1.4 Exit Conditions**

The exit condition for the program is when there were 4 markers placed in a row or there are no valid moves remaining. The 7 exit conditions were tested are the following: horizontal win, vertical win, win-on-flip, win-on-rotate, upper-left-to-lower-right win, upper-right-to-lower-left win, and a tie. The upper right to lower left win condition was not functioning as intended.

**2. Test Plan**

**IDENTIFIER**: 1-TEST-INVALID\_ARG

DESCRIPTION: **Corner case**. When starting the program and an invalid argument is passed, the user shall be informed of proper usage and then the program shall exit.

PRECONDITIONS: None

EXECUTION STEPS:

1. Enter “ruby connect\_four.rb 0” in terminal  
2. Enter “ruby connect\_four.rb hi” in terminal

POSTCONDITIONS: The user is presented with a message on the terminal of proper usage. The program then exits.

**IDENTIFIER**: 2-TEST-VALID-ARG-INVALID\_INT

DESCRIPTION: **Edge case**. When starting the program and an invalid single nonnegative integer is passed, the user shall be informed of proper usage and then the program shall exit.

PRECONDITIONS: None

EXECUTION STEPS:

1. Enter “ruby connect\_four.rb 999999999” in terminal

POSTCONDITIONS: The user is presented with a message on the terminal of proper usage. The program then exits.

**IDENTIFIER**: 3-TEST-VALID-ARG-INVALID\_FLOAT

DESCRIPTION: **Edge case**. When starting the program and an invalid single nonnegative float is passed, the user shall be informed of proper usage and then the program shall exit.

PRECONDITIONS: None

EXECUTION STEPS:

1. Enter “ruby connect\_four.rb 5.6” in terminal

POSTCONDITIONS: The user is presented with a message on the terminal of proper usage. The program then exits.

**IDENTIFIER**: 4-TEST-VALID-ARG-VALID\_INT

DESCRIPTION: When starting the program and a valid single nonnegative integer is passed, the user shall be presented with a game board of x rows and x columns.

PRECONDITIONS: None

EXECUTION STEPS:

1. Enter “ruby connect\_four.rb 5” in terminal

POSTCONDITIONS: The user is presented with a board of 5 rows and 5 columns on the terminal. All spots on the board are blank with the ‘.’ character.

**IDENTIFIER**: 5-TEST-PLAYER-TURN

DESCRIPTION: Players alternate taking turns by placing a marker (X and O) on the board, starting with Player X by typing a valid column in the terminal.

PRECONDITIONS: The program is running and initiated with a 5x5 board.

EXECUTION STEPS:

1. Player X enters “0”  
2. Player O enters “0”

POSTCONDITIONS: The board should display X on position (0,0) and O on position (0,1). Player X should be the next player to make a move.

**IDENTIFIER**: 6-TEST-PLAYER-INVALID\_MOVE

DESCRIPTION: Corner case. Players should not be able to place a marker in an invalid spot or enter a nonvalid command.

PRECONDITIONS: The program is running and initiated with a 5x5 board.

EXECUTION STEPS:

1. Player X enters “0”  
2. Player O enters “9”  
3. Player O enters “blipblop”

POSTCONDITIONS: The terminal should display to Player O the invalid command, the list of valid moves, and the game board with a marker X on position (0,0) for each invalid move. Player O should be able to try to make another move.

**IDENTIFIER**: 7-TEST-PLAYER-INVALID\_MOVE\_FULL

DESCRIPTION: Players should not be able to place a marker in a full column.

PRECONDITIONS: The program is running and initiated with a 4x4 board.

EXECUTION STEPS:

1. Player X enters “0”  
2. Player O enters “0”  
3. Player X enters “0”  
4. Player O enters “0”

POSTCONDITIONS: The terminal should display to Player O the invalid move, the list of valid moves, and the game board with a marker X on positions (0,0) and (0,2) and a marker O on position (0,1). Player O should be able to try to make another move.

**IDENTIFIER**: 8-TEST-MOVE-VALID\_COLUMN

DESCRIPTION: Upon selecting a column number, a piece corresponding to the current player will be dropped and fall to the lowest spot that is still blank.

PRECONDITIONS: The program is running and initiated with a 4x4 board.

EXECUTION STEPS:

1. Player X enters “0”  
 2. Player O enters “1”

POSTCONDITIONS: The terminal should display the game board with a marker ‘X’ on position (0,0) and a marker ‘O’ on position (1,0).

**IDENTIFIER**: 9-TEST-MOVE-FLIP\_BOARD

DESCRIPTION: When a player types “flip” as their move, the board should flip 180 degrees and all of the pieces currently on the board should flip in its configuration.

PRECONDITIONS: The program is running and initiated with a 4x4 board. Player X has markers on positions (0,0) and (1,0) and Player O has markers on positions (0,1) and (1,1). It is Player X’s turn.

EXECUTION STEPS:

1. Player X types “flip”  
 2. Press enter

POSTCONDITIONS: Player X has markers on position (0,1) and (1,1) and Player O has markers on position (0,0) and (1,0).

**IDENTIFIER**: 10-TEST-MOVE-FLIP-WIN

DESCRIPTION: When a player types “flip” and a connect four is created this way, the game is won by the player flipping the board regardless of the type of marker.

PRECONDITIONS: The program is running and initiated with an 8x8 board. Player X has markers on positions (0,0), (0,1), (1,1), (2,1), and (3,2). Player O has markers on positions (1,0), (2,0), (3,0), and (3,1).

EXECUTION STEPS:

1. Player O types “flip”  
 2. Press enter

POSTCONDITIONS: Player X has markers on positions (0,0), (1,0), (2,0), (3,0), and (0,1). Player O has markers on positions (1,1), (2,1), (3,1), and (3,2). The program indicates Player O has won the game and exits.

**IDENTIFIER**: 11-TEST-MOVE-CAPS-ROT\_BOARD

DESCRIPTION: When a player types “ROT” as their move, the board should rotate clockwise 90 degrees. The game markers fall to the new “floor” of the game board.

PRECONDITIONS: The program is running and initiated with a 4x4 board. Player X has markers on positions (0,0) and (0,2). Player O has markers on positions (0,1) and (0,3).

EXECUTION STEPS:

1. Player X types “ROT”  
 2. Press enter

POSTCONDITIONS: Player X has markers on positions (0,0) and (2,0) and player O has markers on position (1,0) and (3,0).

**IDENTIFIER**: 12-TEST-MOVE-ROT-WIN

DESCRIPTION: When a player types “rot” and a connect four is created this way, the game is won by the player rotating the board regardless of the type of marker.

PRECONDITIONS: The program is running and initiated with an 8x8 board. Player X has markers on positions (0,0), (0,1), (0,2), (0,3). Player O has markers on positions (2,0), (2,1), (3,2), and (1,3).

EXECUTION STEPS:

1. Player X types “rot”  
2. Press enter

POSTCONDITIONS: Player X has markers on positions (0,1), (1,1), (2,1), (0,2), and (0,1). Player O has markers on positions (0,0), (1,0), (2,0), and (3,0). The program indicates Player X has won the game and exits.

**IDENTIFIER**: 13-TEST-WINNER-HORIZONTAL

DESCRIPTION: When any four markers of the same type is placed horizontally in consecutive order anywhere on the game board, the player of that marker type wins the game. The program indicates the winning player and exits the game.

PRECONDITIONS: The program is running and initiated with a 4x4 board.

EXECUTION STEPS:

1. Player X types 0  
 2. Press enter  
 3. Player O types 0  
 4. Press enter  
 5. Player X types 1  
 6. Press enter  
 7. Player O types 1  
 8. Press enter  
 9. Player X types 2  
 10. Press enter  
 11. Player O types 2  
 12. Press enter  
 13. Player X types 3  
 14. Press enter

POSTCONDITIONS: The program indicates Player X has won and exits.

**IDENTIFIER**: 14-TEST-WINNER-VERTICAL

DESCRIPTION: When any four markers of the same type is placed vertically in consecutive order anywhere on the game board, the player of that marker type wins the game. The program indicates the winning player and exits the game.

PRECONDITIONS: The program is running and initiated with a 4x4 board.

EXECUTION STEPS:

1. Player X types 0  
 2. Press enter  
 3. Player O types 1

4. Press enter  
5. Player X types 0  
6. Press enter  
7. Player O types 1  
8. Press enter  
9. Player X types 0  
10. Press enter  
11. Player O types 1  
12. Press enter  
13. Player X types 0  
14. Press enter

POSTCONDITIONS: The program indicates Player X has won and exits.

**IDENTIFIER**: 15-TEST-WINNER-DIAGONAL

DESCRIPTION: When any four markers of the same type is placed diagonally in consecutive order anywhere on the game board, the player of that marker type wins the game. The program indicates the winning player and exits the game.

PRECONDITIONS: The program is running and initiated with a 4x4 board.

EXECUTION STEPS:

1. Player X types 0  
2. Press enter  
3. Player O types 1  
4. Press enter  
5. Player X types 1  
6. Press enter  
7. Player O types 2  
8. Press enter  
9. Player X types 3  
10. Press enter  
11. Player O types 2  
12. Press enter  
13. Player X types 2  
14. Press enter  
15. Player O types 3  
16. Press enter  
17. Player X types 0  
18. Press enter  
19. Player O types 3  
20. Press enter  
21. Player X types 3  
22. Press enter

POSTCONDITIONS: The program indicates Player X has won and exits.

**IDENTIFIER**: 16-TEST-COLUMN-OVERNINE-DISPLAY

DESCRIPTION: When the game is initialized with a board size of over nine spaces. The board will represent the spots over nine spaces by the last digit of their number.

PRECONDITIONS: None

EXECUTION STEPS:

1. Type “ruby connect\_four.rb 19” in terminal  
2. Press enter

POSTCONDITIONS: The column numbers over nine should display only the last digit of the respective numbers.

**IDENTIFIER**: 17-TEST-COLUMN-OVERNINE-MOVE

DESCRIPTION: When the game is initialized with a board size of over nine spaces, submitting a move over nine shall display the marker in the appropriate spot.

PRECONDITIONS: The program is running and initiated with a 19x19 board.

EXECUTION STEPS:

1. Player X types 18  
2. Press enter

POSTCONDITIONS: The board should display a marker X on position (18,0)

**IDENTIFIER**: 18-TEST-COLUMN-OVERNINE-INVALID-MOVE

DESCRIPTION: When the game is initialized with a board size of over nine spaces, players should not be able to play a move beyond the valid spots on the board.

PRECONDITIONS: The program is running and initiated with a 19x19 board.

EXECUTION STEPS:

1. Player X types 20  
2. Press enter

POSTCONDITIONS: The terminal should display to Player X the invalid command and the list of valid moves. Player X should be able to try to make another move.

**3.** **Traceability Matrix**

**REQ\_1:** 1-TEST-INVALID\_ARG, 2-TEST-VALID-ARG-INVALID\_INT, 3-TEST-VALID-ARG-INVALID\_FLOAT **REQ\_2:** 4-TEST-VALID-ARG-VALID\_INT **REQ\_3:** 5-TEST-PLAYER-TURN **REQ\_4:** 6-TEST-PLAYER-INVALID\_MOVE **REQ\_5:** 7-TEST-PLAYER-INVALID\_MOVE\_FULL, 8-TEST-MOVE-VALID\_COLUMN **REQ\_6:** 9-TEST-MOVE-FLIP\_BOARD, 10-TEST-MOVE-FLIP-WIN **REQ\_7:** 11-TEST-MOVE-CAP-ROT\_BOARD, 12-TEST-MOVE-ROT-WIN **REQ\_8:** 13-TEST-WINNER-HORIZONTAL, 14-TEST-WINNER-VERTICAL, 15-TEST-WINNER-DIAGONAL **REQ\_9:** 16-TEST-COLUMN-OVERNINE-DISPLAY, 17-TEST-COLUMN-OVERNINE-MOVE, 18-TEST-COLUMN-OVERNINE-INVALID-MOVE

Table 1 - Requirements (Rows) matched with its Test Case Identifier (Columns)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| REQ\_1 | **X** | **X** | **X** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| REQ\_2 |  |  |  | **X** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| REQ\_3 |  |  |  |  | **X** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| REQ\_4 |  |  |  |  |  | **X** |  |  |  |  |  |  |  |  |  |  |  |  |
| REQ\_5 |  |  |  |  |  |  | **X** | **X** |  |  |  |  |  |  |  |  |  |  |
| REQ\_6 |  |  |  |  |  |  |  |  | **X** | **X** |  |  |  |  |  |  |  |  |
| REQ\_7 |  |  |  |  |  |  |  |  |  |  | **X** | **X** |  |  |  |  |  |  |
| REQ\_8 |  |  |  |  |  |  |  |  |  |  |  |  | **X** | **X** | **X** |  |  |  |
| REQ\_9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **X** | **X** | **X** |

**4. Defects Found and Fixed**

**SUMMARY:** The program will take a floating-point value as a valid command-line argument

**DESCRIPTION:** When a floating-point value is used as a command-line argument for size, the program will initialize a board with the value left of the decimal of the floating-point value. Only non-negative integers should be allowed for the command-line argument and all others should be rejected.

**REPRODUCTION STEPS:**

* + - 1. Enter “ruby connect\_four.rb 5.6” in terminal

**EXPECTED BEHAVIOR:** The program should terminate with a message denoting that only non-negative integers are allowed.

**OBSERVED BEHAVIOR:** The board will initialize with the value left of the decimal of the floating-point value.

**SUMMARY:**  The program will not recognize a diagonal win going from the upper right to lower left.

**DESCRIPTION:** The program is not recognizing the win condition when there are 4 markers connected diagonally from the upper right to lower left.

**REPRODUCTION STEPS:**

Start program with integer greater or equal to 4

Place the same type of marker diagonally from lower left to upper right

**EXPECTED BEHAVIOR:** The program will exist saying “Player \_ won!” The “\_” denoting whether player X or O won.

**OBSERVED BEHAVIOR:** The program will continue prompting the opposing player to enter a move.

**SUMMARY:** When players use move “ROT” the program will add a marker to the board.

**DESCRIPTION:** The program requires player moves to be case-insensitive but when using the move “ROT” the program will add an extra marker instead of rotating.

**REPRODUCTION STEPS:**

* + - 1. Start the program with a valid size
      2. Either player can enter move “ROT” in all caps at any point

**EXPECTED BEHAVIOR:** The board will rotate and prompt the next player for their move.

**OBSERVED BEHAVIOR:** There will be a marker placed from the respective player entering the move in the left most column. If the left most column is full then the program will crash with a Runtime Error “Could not drop piece in column 0.”