CS1632 – Deliverable 1: Test Plan and Traceability Matrix

Project: Super Connect Four!

Daesang Yoon (yds725) and Chris ??

Introduction:

Test Cases:

Identifer: Test\_Non-Negative\_Integer

Description: Determine if the program accepts non-negative integer as argument from the command line and launches successfully.

Preconditions: The program has not yet started. User is at command line.

Execution Steps: Pass non-negative integer 7 as argument. Use command ruby connect\_four.rb 7 to run the program.

Postconditions: The program launches and displays gameboard. The program built the board is 7(columns) x 7(rows) just like below:

0123456

.......

.......

.......

.......

.......

.......

.......

Player X, enter move >

Identifer: Test\_Non-Numeric\_Value

Description: Test for the program exits and display proper direction for user when a non-numeric value is passed as argument at the command line.

Preconditions: The program has not yet started. User is at command line.

Execution Steps: Pass non-numeric value chris as argument. Use command ruby connect\_four.rb chris to run the program.

Postconditions: The program exits. The program displays and explains proper direction to user.

Identifer: Test\_Negative\_Integer

Description: Test for the program exits and display proper direction for user when a negative integer is passed as argument at the command line.

Preconditions: The program has not yet started. User is at command line.

Execution Steps: Pass negative integer -7 as argument. Use command ruby connect\_four.rb -7 to run the program

Postconditions: The program exits. The program displays and explains proper direction to user.

Identifer: Test\_Non-Integer\_Argument

Description: Test for the program exits and display proper direction for user when a non-integer numeric value (float and double) is passed as argument at the command line.

Preconditions: The program has not yet started. User is at command line.

Execution Steps: Pass non-integer 7.92 as argument. Use command ruby connect\_four.rb 7.92 to run the program.

Postconditions: The program exits. The program displays and explains proper direction to user.

Identifer: Test\_Empty\_Argument

Description: Test for the program exits and display proper direction for user when nothing (empty string) is passed as argument at the command line.

Preconditions: The program has not yet started. User is at command line.

Execution Steps: Pass no argument. Use command ruby connect\_four.rb to run the program.

Postconditions: The program exits. The program built displays and explains proper direction to user.

Identifer: Test\_Valid\_Integer

Description: When a valid integer ‘x’ is entered as argument, the program should construct a game board of x rows and x columns and initially fill board with ‘.’ character representing blank space.

Preconditions: The program has not yet started. User is at command line.

Execution Steps: Pass non-negative integer 7 as argument. Use command ruby connect\_four.rb 7 to run the program.

Postconditions: The program constructs 7 x 7 game board which is filled with 7 x 7 ‘.’ characters. Top of the board, integers in the range of 0 … 7-1 (0123456) are placed as column labels.

Identifer: Test\_Size\_Less\_Than\_Four (Edge Case)

Description: ‘Super Connect Four’ requires each player to connect 4 checkers of same type either horizontally, vertically or diagonally. A gameboard that has number of rows and columns less than 4 should not be valid. Edge case.

Preconditions: The program has not yet started. User is at command line.

Execution Steps: Pass non-negative integer 2 as argument. Use command ruby connect\_four.rb 2 to run the program.

Postconditions: The program constructs 2 x 2 game board which is filled with 2 x 2 ‘.’ characters. Top of the board, integers in the range of 0 … 2-1 (01) are placed as column labels.

Identifer: Test\_First\_Turn for Player X

Description: Player X gets the first turn when a new game is started.

Preconditions: The program has not yet started. User is at command line.

Execution Steps: Pass non-negative integer 7 as argument. Use command ruby connect\_four.rb 7 to run the program. Then, Enter the input ‘5’.

Postconditions: An ‘X’ checker which represents player X’s checker is put into in the column 5 of the board.

Identifer: Test\_Switching\_Turns

Description: Test for whether turns alternate between Player X and Player O after each player makes each move.

Preconditions: The program has started. The gameboard size of 7 has been constructed.

Execution Steps:

1. Player X, enter move > 5

2. Player O, enter move > 0

3. Player X, enter move > 2

4. Player O, enter move > 1

Postconditions: Turns alternate between Player X and Player O after each player enters move. Checker ‘X’s are dropped into column 2 and 5. Checker ‘O’s are dropped into column 0 and 1.

. . . . . . .

. . . . . . .

. . . . . . .

. . . . . . .

. . . . . . .

. . . . . . .

O O X . . X .

Identifer: Test\_Non-Existing\_Column

Description: An error message should be displayed when the player enters a column number which does not exist within the gameboard.

Preconditions: The program has started. The gameboard size of 7 has been constructed. It is Player X’s turn to enter a move.

Execution Steps:

1. Player X, enter move > 8

Postconditions: The following error message is displayed and the game gives another opportunity to player to enter a valid input without switching turns or changing players.

8 is an invalid move.

Please enter one of the following:

1. A valid column number

2. 'flip' to flip the board

3. 'rot' to rotate the board

Identifer: Test\_Full\_Column (Edge Case)

Description: An error message should be displayed when the player enters a column number which is already full. Edge Case.

Preconditions: The program has started. The gameboard size of 7 has been constructed. At column 3, there are already 7 checkers are placed. Column 3 is now full.

Execution Steps:

1. Player O, enter move > 3

Postconditions: The following error message is displayed and the game gives another opportunity to player to enter a valid input without switching turns or changing players:

3 is an invalid move.

Please enter one of the following:

1. A valid column number

2. 'flip' to flip the board

3. 'rot' to rotate the board

Identifer: Test\_Invalid\_String,

Description: Players can enter ‘flip’ and ‘rot’ to flip and rotate the board during their turn. These commands are case-insensitive. All other strings other than non-negative integer column numbers are invalid.

Preconditions: The program has started. The gameboard size of 7 has been constructed. It is Player X’s turn to enter a move.

Execution Steps:

1. Player X, enter move > flipchris

Postconditions: The following error message is displayed and the game gives another opportunity to player to enter a valid input without switching turns or changing players.

flipchris is an invalid move.

Please enter one of the following:

1. A valid column number

2. 'flip' to flip the board

3. 'rot' to rotate the board

Identifer: Test\_Case\_Sensitivity\_Flip (Edge Case)

Description: Players can enter ‘flip’ with any combination of lower case and upper case to flip the board during their turn. For example, players can enter ‘fLip’, ‘FLIP’, ‘FLIp’, etc.

Preconditions: The program has started. The gameboard size of 7 has been constructed. It is Player X’s turn to enter a move.

Execution Steps:

1. Player X, enter move > 2

2. Player O, enter move > 2

3. Player X, enter move > FLIp

Postconditions: The gameboard has been successfully flipped.

0123456

.......

.......

.......

.......

.......

..X....

..O....

Player O, enter move >

Identifer: Test\_Case\_Sensitivity\_Rot (Edge Case)

Description: Players can enter ‘rot’ with any combination of lower case and upper case to rotate the board 90 degrees to the right during their turn. For example, players can enter ‘ROt’, ‘rOt’, ‘ROT’, ‘RoT’, etc.

Preconditions: The program has started. The gameboard size of 7 has been constructed. It is Player X’s turn to enter a move.

Execution Steps:

1. Player X, enter move > 2

2. Player O, enter move > 3

3. Player X, enter move > rOt

Postconditions: The gameboard has been successfully rotated 90 degrees to the right:

0123456

.......

.......

.......

.......

.......

X......

O......

Player O, enter move >

Identifer: Test\_Column\_Checker\_Function

Description: Upon selecting a column number, a piece corresponding to the current player is dropped and take up the lowest spot that is still blank.

Preconditions: The program has started. The gameboard size of 7 has been constructed. Player X has dropped its piece in column 2.

Execution Steps: Player O makes a move in column 2.

1. Player O, enter move > 2

Postconditions: ‘O’ piece appear in column 2, right at the top of ‘X’ piece which was already placed in very bottom of column 2.

Identifer: Test\_Flip

Description: ‘flip’ command causes the board to flip upside down and all pieces fall into the floor or bottom of the gameboard.

Preconditions: The program has started. The gameboard size of 7 has been constructed.

1. Player X, enter move > 3

2. Player O, enter move > 3

Execution Steps: 1. Player X, enter move > flip

Postconditions: The gameboard has been successfully flipped. All pieces fall into the bottom of the gameboard.

0123456

.......

.......

.......

.......

.......

...X...

...O...

Player O, enter move >

Identifer: Test\_Flip\_Winner

Description: ‘flip’ command causes the board to flip upside down and all pieces fall into the floor or the bottom of the gameboard. If a 4 pieces of same type are connected in this way, the game is won by the player flipping the board (even if it is the other player's pieces).

Preconditions: The program has started. The gameboard size of 7 has been constructed. Current State of gameboard:

0123456

.......

.......

.......

...O...

...X...

...X...

.X.XOOO

Execution Steps: 1. Player X, enter move > flip

Postconditions: The gameboard has been successfully flipped. All pieces fall into the bottom of the gameboard. The player X won the game even though 4 ‘O’ pieces are connected. The game finishes.

0123456

.......

.......

.......

...X...

...X...

...X...

.X.OOOO

Player X won!

Identifer: Test\_Rotate

Description: ‘rot’ command causes the board to rotate 90 degrees to the right and all pieces fall into the new floor or bottom of the gameboard.

Preconditions: The program has started. The gameboard size of 7 has been constructed.

Execution Steps:

1. Player X, enter move > 2

2. Player O, enter move > 3

3. Player X, enter move > 2

4. Player O, enter move > 4

5. Player X, enter move > 5

6. Player O, enter move > 6

0123456

.......

.......

.......

.......

.......

..X....

..XOOXO

7. Player X, enter move > rot

Postconditions: The gameboard has been successfully rotated. All pieces fall into the new bottom of the gameboard.

0123456

.......

.......

X......

O......

O......

X......

OX.....

Identifer: Test\_Rotate\_Winner

Description: Tests for whether the program actually detects a win after a player rotates the gameboard, and a player who rotates the board becomes the winner even if the other player’s 4 pieces are connected.

Preconditions: The program has started. The gameboard size of 7 has been constructed.

Execution Steps:

1. Player X, enter move > 6

2. Player O, enter move > 6

3. Player X, enter move > 5

4. Player O, enter move > 5

5. Player X, enter move > 4

6. Player O, enter move > 4

7. Player X, enter move > 1

0123456

.......

.......

.......

.......

.......

....OOO

.X..XXX

8. Player O, enter move > rot

Postconditions: The gameboard will be rotated 90 degrees to the right. All pieces will fall into the new bottom of the gameboard. The player O will win the game even though 4 ‘X’ pieces are connected. The game finishes with message showing player O has won the game.

0123456

.......

.......

.......

X......

XO.....

XO.....

XO.....

Identifer: Test\_Connect-Four\_Vertical

Description: The first player to make a ‘connect four’, four checkers of same type horizontally, vertically, or diagonally wins the game. The game announces the winner and exit.

Preconditions: The program has started. The gameboard size of 7 has been constructed.

Execution Steps:

1. Player X, enter move > 5

2. Player O, enter move > 6

3. Player X, enter move > 5

4. Player O, enter move > 6

5. Player X, enter move > 5

6. Player O, enter move > 6

7. Player X, enter move > 5

Postconditions: 4 ‘X’ pieces are successfully connected vertically. A message that shows player X is winner is displayed and the game finishes.

0123456

.......

.......

.......

.....X.

.....XO

.....XO

.....XO

Player X won!

Identifer: Test\_Connect-Four\_Horizontal

Description: The first player to make a ‘connect four’, four checkers of same type horizontally, vertically, or diagonally wins the game. The game announces the winner and exit.

Preconditions: The program has started. The gameboard size of 7 has been constructed.

Execution Steps:

1. Player X, enter move > 6

2. Player O, enter move > 6

3. Player X, enter move > 5

4. Player O, enter move > 5

5. Player X, enter move > 4

6. Player O, enter move > 4

7. Player X, enter move > 3

Postconditions: 4 ‘X’ pieces are successfully connected horizontally. A message that shows player X is winner is displayed and the game finishes.

0123456

.......

.......

.......

.......

.......

....OOO

...XXXX

Player X won!

Identifer: Test\_Connect-Four\_Diagonal

Description: The first player to make a ‘connect four’, four checkers of same type horizontally, vertically, or diagonally wins the game. The game announces the winner and exit.

Preconditions: The program has started. The gameboard size of 7 has been constructed.

Execution Steps:

1. Player X, enter move > 6

2. Player O, enter move > 5

3. Player X, enter move > 5

4. Player O, enter move > 4

5. Player X, enter move > 4

6. Player O, enter move > 3

7. Player X, enter move > 4

8. Player O, enter move > 3

9. Player X, enter move > 3

10. Player O, enter move > 1

11. Player X, enter move > 3

Postconditions: 4 ‘X’ pieces are successfully connected diagonally. A message that shows player X is winner is displayed and the game finishes.

0123456

.......

.......

.......

...X...

...XX..

...OXX.

.O.OOOX

Player X won!

Identifer: Test\_Diplay\_Last\_Digit

Description: The gameboard indicates the column numbers with the last digit of their number. For example, column 17 will be displayed as ‘7’.

Preconditions: The program has not yet started. User is at the command line.

Execution Steps: Pass non-negative integer 18 as argument. Use command ruby connect\_four.rb 18 to run the program.

Postconditions:

The program launches and displays gameboard. The program built the board that is 18 columns X 18 rows. All column numbers are labeled as last digit of their number like the following:

012345678901234567

..................

..................

..................

(Omitted some lines)

..................

..................

Identifer: Test\_Input\_Last\_Digit

Description: Player can still drop a checker into any column by entering actual full number value for that column. For example, despite column 17 is being displayed as column ‘7’, players can still enter ‘17’ to make a move in that column.

Preconditions: The program has started. The gameboard of size 18 X 18 is created:

012345678901234567

..................

..................

..................

(Omitted some lines)

..................

..................

Player X, enter move >

Execution Steps: Player X drops a checker in column 16 by entering ‘16’ in command line.

1. Player X, enter move > 16

Postconditions:

Player X successfully drops a checker in the column 16. The board will be as following state:

012345678901234567

..................

..................

..................

(Omitted some lines)

..................

................X.

Player O, enter move >

Traceability Matrix

Requirement 1 : Test\_Non-Negative\_Integer, Test\_Negative\_Integer, Test\_Non-Numeric\_Value, Test\_Non-Integer\_Argument, Test\_Empty\_Argument

Requirement 2: Test\_Valid\_Integer, Test\_Size\_Less\_Than\_Four (Edge Case)

Requirement 3: Test\_First\_Turn for Player X, Test\_Switching\_Turns

Requirement 4: Test\_Non-Existing\_Column, Test\_Full\_Column (Edge Case), Test\_Invalid\_String, Test\_Case\_Sensitivity\_Flip, (Edge Case) Test\_Case\_Sensitivity\_Rot (Edge Case)

Requirement 5:Test\_Column\_Checker\_Function

Requirement 6:Test\_Flip, Test\_Flip\_Winner

Requirement 7:Test\_Rotate, Test\_Rotate\_Winner

Requirement 8:Test\_Connect-Four\_Vertical, Test\_Connect-Four\_Horizontal, Test\_Connect-Four\_Diagonal

Requirement 9: Test\_Diplay\_Last\_Digit, Test\_Input\_Last\_Digit

Test Results

Test\_Non-Negative\_Integer : Pass

Test\_Negative\_Integer: Pass

Test\_Non-Numeric\_Value: Pass

Test\_Non-Integer\_Argument: Fail

Test\_Empty\_Argument: Pass

Test\_Valid\_Integer: Pass

Test\_Size\_Less\_Than\_Four (Edge Case): Pass

Test\_First\_Turn for Player X: Pass

Test\_Switching\_Turns: Pass

Test\_Non-Existing\_Column: Pass

Test\_Full\_Column (Edge Case): Pass

Test\_Invalid\_String: Pass

Test\_Case\_Sensitivity\_Flip (Edge Case): Pass

Test\_Case\_Sensitivity\_Rot (Edge Case): Fail

Test\_Column\_Checker\_Function: Pass

Test\_Flip: Pass

Test\_Flip\_Winner: Pass

Test\_Rotate: Pass

Test\_Rotate\_Winner: Fail

Test\_Connect-Four\_Vertical: Pass

Test\_Connect-Four\_Horizontal: Pass

Test\_Connect-Four\_Diagonal: Pass

Test\_Diplay\_Last\_Digit: Fail

Test\_Input\_Last\_Digit: Pass

Report Defects

1)

Summary: Entering ‘rot’ command with combination of mixed lower and upper case can cause extra piece to be placed into the board and not rotate.

Description: When user enter ‘rot’ as mixed case (‘ROt’, ‘rOt’, ‘ROT’, ‘RoT’), the board should just rotate 90 degrees but it does not rotate instead it drops one piece in column 0. Defect was found in Test\_Case\_Sensivity\_Rot.

Reproduction Steps:

1. ruby connect\_four.rb 7

2. Player X, enter move > 2

3. Player O, enter move > 3

4. Player X, enter move > rOt

Expected Behavior:

The gameboard should be in following state:

0123456

.......

.......

.......

.......

.......

X......

O......

Player O, enter move >

Observed Behavior: The board is not rotated and an extra piece is dropped into column 0.

0123456

.......

.......

.......

.......

.......

.......

X.XO...

Player O, enter move >

2)

Summary: The gameboard should display column numbers as their last digit number.

Description: If a gameboard is created with size greater than 10, then column numbers should be displayed as their last digit. Ex) 10 🡪 0, 11 🡪 1, 12 🡪 2, etc. Defect was found in Test\_Display\_Last\_Digit test case.

Reproduction Steps:

1. ruby connect\_four.rb 18

Expected Behavior:

The gameboard should display column numbers as their last digit. We expect gameboard to be following:

012345678901234567

..................

..................

..................

(Omitted some lines)

..................

Observed Behavior:

Starting from 10, the gameboard fails to display column numbers as their last digit. The actual result:

01234567891011121314151617

..................

..................

..................

..................

..................

..................

..................

..................

..................

Omitted Some lines

3)

Summary: The gameboard should display a proper usage and direction to user and exit.

Description: If user pass non-integer value 7.5 (float or double value), the program should display a proper usage and direction to user and exit. The defect was found in Test\_Non-Integer\_Argument test case.

Reproduction Steps:

1. ruby connect\_four.rb 7.5

Expected Behavior:

The program should display a proper usage to user and exit successfully. Like following message:

Usage:

ruby connect\_four.rb \*x\*

\*x\* should be a nonnegative integer

Observed Behavior:

Instead of displaying proper usage and explaining direction to user, the program constructs gameboard size of 7 columns x 7 rows. The program treats 7.5 as Integer value 7.

0123456

.......

.......

.......

.......

.......

.......

.......

Player X, enter move >