CSCI1120 Grading Scheme
Assignment 5 Twin Knights Tour

2. Test cases

2.1 Constructor

Test the public constructor function for the class.

Case	Constructor	Outputs	Description

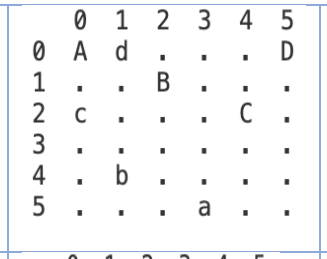
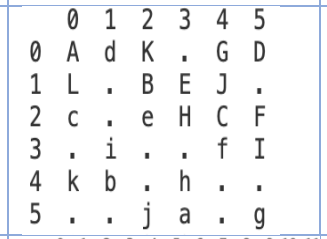
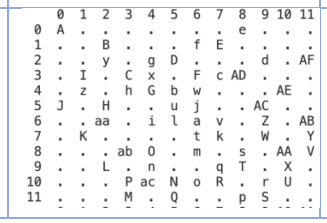
1	TwinKnightsTour game(0,0,3,3);	None	**Check:**
			board[0][0] = "A";
			board[3][3] = "a";
			for other i, j : board[i][j]='.'
			posR1 = 0;
			posC1 = 0;
			posR2 = 3;
			posC2 = 3;
			steps1 = 0;
			steps2 = 0;
			consec1 = 0;
			consec2 = 0;

2	TwinKnightsTour game(5,0,1,3);	None	**Check:**
			board[5][0] = "A";
			board[1][3] = "a";
			for other i, j : board[i][j]='.'
			posR1 = 5;
			posC1 = 0;
			posR2 = 1;
			posC2 = 3;
			steps1 = 0;
			steps2 = 0;
			consec1 = 0;
			consec2 = 0;

2.2 Function Unit Test for print

Test the print() member function.

Case	Board content	Std out	Description

1		Output: 0 1 2 3 4 5 0 A # . . . @ 1 . . B . . . 2 c . . . C . 3 . . . . . . 4 . b . . . . 5 . . . a . .	Note the spacing!
2		0 1 2 3 4 5 0 A d K . G D 1 @ . B E J . 2 c . e H C F 3 . i . . f I 4 # b . h . . 5 . . j a . g	Note the spacing !
3			A board with size 12*12.
4			Note the spacing and 2-width letters!

2.3 Function Unit Test for isValid()

Test the isValid() member function.

Case	Inputs	Outputs	Description
1	Board: 0 1 2 3 4 5 0 A # . . . @ 1 . . B . . . 2 c . . . C . 3 . . . . . . 4 . b . . . . 5 . . . a . . Knight=@ r=10 c=-1 posR1=0 posC1=5 posR2=0 posC2=1 consec1=0	False	Out of board

	consec2=1		

2	Board:
	`  0 1 2 3 4 5`
	`0 A # . . . @`
	`1 . . B . . .`
	`2 c . . . C .`
	`3 . . . . . .`
	`4 . b . . . .`
	`5 . . . a . .`
	Knight='*'
	r=2
	c=2
	posR1=0
	posC1=5
	posR2=0
	posC2=1
	consec1=0
	consec2=1
		False	The wrong knight character

3	Board:
	`  0 1 2 3 4 5`
	`0 A # . . . @`
	`1 . . B . . .`
	`2 c . . . C .`
	`3 . . . . . .`
	`4 . b . . . .`
	`5 . . . a . .`
	Knight='#'
	r=0
	c=3
	posR1=0
	posC1=5
	posR2=0
	posC2=1
	consec1=0
	consec2=1
		False	The move is not L-shaped.

4	Board:	False	The move is the third consecutive move for "@" (1st move: @ 1 3 2nd move: @ 2 5)

	0 1 2 3 4 5 0 A # . . . D 1 . . B E . . 2 c . . . C @ 3 . . . . . . 4 . b . . . . 5 . . . a . . Knight='@' r=4 c=4 posR1=2 posC1=5 posR2=0 posC2=1 consec1=2 consec2=0		
5	Board: 0 1 2 3 4 5 6 7 8 9 10 11 0 A . . . . e . . . 1 . . B . . . f E . . # . 2 . . . . g D . . o d . . 3 . I . C . . F c . . . . 4 . . . h G b . . . n . . 5 J . H . . . j m . . . . 6 . . . . i l a . . . . . 7 . K . . R . . k . . . . 8 . . S . O . . . . 9 @ . L . . Q . . . . . . 10 . . . P . N . . . . . . 11 . . . M . . . . Knight='#' r=2 c=8 posR1=9 posC1=0 posR2=1 posC2=10 consec1=2 consec2=0	False	Try to move to a visited position
6	Board:	True	A valid case.

	0 1 2 3 4 5 6 7 8 9 10 11 0 A . . . . e . . . 1 . . B . . . f E . . p . 2 . . . . g D . r o d . . 3 . I . C . . F c . q . . 4 . . . h G b . . # n . . 5 J . H . . . j m . . . . 6 . . . . i l a . . . . . 7 . K . . R . . k . . . . 8 . . S . O . . . . 9 T . L . . Q . . . . . . 10 . . . P . N . . . . . . 11 . @ . M . . . . Knight='#' r=5 c=10 posR1=11 posC1=1 posR2=4 posC2=8 consec1=0 consec2=1		

2.4 Function Unit Test for hasMoreMoves()

Test the hasMoreMoves() member function.

Case	Board	Outputs	Description
1	Board: 0 1 2 3 4 5 0 A d K . G D 1 @ . B E J . 2 c . e H C F 3 . i . . f I 4 # b . h . . 5 . . j a . g consec1=0 consec2=2 posR1=1 posC1=0 posR2=4 posC2=0	False	@ has no valid moves. # has made two consecutive moves.
2	Board:	True	@ has no valid moves. # still has two potential moves.

	0 1 2 3 4 5 0 A d . . G D 1 . . B E J . 2 c # e H C F 3 . i . K f I 4 k b . h . @ 5 . . j a . g consec1=0 consec2=1 posR1=4 posC1=5 posR2=2 posC2=2		
3	Board: 0 1 2 3 4 5 0 A d . . G D 1 # . B E J . 2 c L e H C F 3 . i . K f I 4 @ b . h . . 5 . . . a . g consec1=2 consec2=0 posR1=4 posC1=0 posR2=1 posC2=0	True	@ has made two consecutive moves. # has only one valid move.
4	Board: 0 1 2 3 4 5 6 7 8 9 10 11 0 A . . . . e . . . 1 . . B . . . f E . . p . 2 . . . . g D . r o d . . 3 . I . C . . F c . q . . 4 . . . h G b . . # n . . 5 J . H . . . j m . . . . 6 . . . . i l a . . . . . 7 . K . . R . . k . . . . 8 . . S . O . . . . 9 T . L . . Q . . . . . . 10 . . . P . N . . . . . . 11 . @ . M	True	@ has no valid moves. # still has several potential and valid moves.

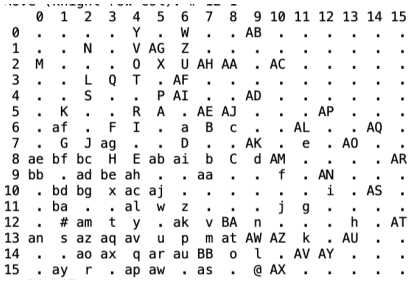
	consec1=0 consec2=1 posR1=11 posC1=1 posR2=4 posC2=8		

2.5 Function Unit Test for move()

Test the move() member function.

Case	Inputs	Outputs	Description
1	Board: 0 1 2 3 4 5 0 A # . . . @ 1 . . B . . . 2 c . . . C . 3 . . . . . . 4 . b . . . . 5 . . . a . . Knight='@' r=1 c=3 consec1=0 consec2=1 posR1=0 posC1=5 posR2=0 posC2=1 steps1=3 steps2=3	True	Need to check the data members. Consec1=1 Consec2=0 posR1=1 posC1=3 Board[1][3]=='E'
2	Board:	False	Not a valid move. No update on data members!

	0 1 2 3 4 5 0 A # . . . D 1 . . B @ . . 2 c . . . C . 3 . . . . . . 4 . b . . . . 5 . . . a . . Knight='@' r=1 c=4 consec1=1 consec2=0 posR1=1 posC1=3 posR2=0 posC2=1 steps1=4 steps2=3		
3	Board: 0 1 2 3 4 5 6 7 8 9 10 11 0 A . . . . e . . . 1 . . B . . . f E . . . . 2 . . . . g D . . . d . . 3 . I . C . . F c . . . . 4 . . . h G b # . . . . . 5 J . H . . u j . . . . . 6 . . . . i l a v . @ . . 7 . K . . . . t k . W . Y 8 . . . . O . m . s . . V 9 . . L . n . . q T . X . 10 . . . P . N o R . r U . 11 . . . M . Q . . p S . . Knight='@' r=8 c=10 consec1=0 consec2=2 posR1=6 posC1=9 posR2=4 posC2=6 steps1=26	True	.Need to check the data members. Consec1=1 Consec2=0 posR1=8 posC1=10 Board[8][10]='AA'

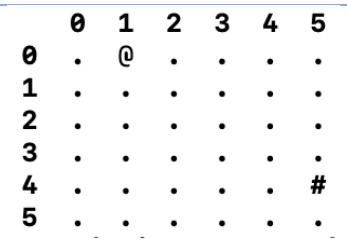
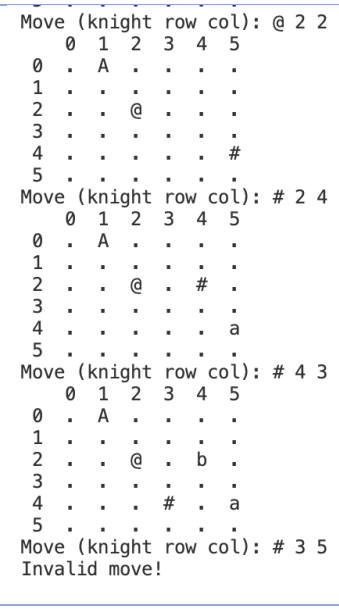
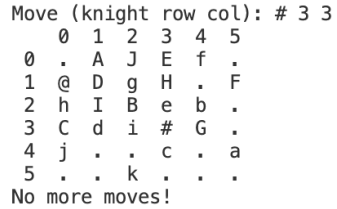
	steps2=23		
4	Board:  Knight='#' r=10 c=0 posR1: 15 posC1: 9 posR2: 12 posR2: 1 steps1: 54 steps2: 59 consec1: 0 consec2: 1	True	Need to check the data members. posR1: 15 posC1: 9 posR2: 10 posR2: 0 consec1: 0 consec2: 2 Board[10][0]='bi'

2.6 Gameflow Test

Here we use 2 test flows to test the implementation of gameflow.

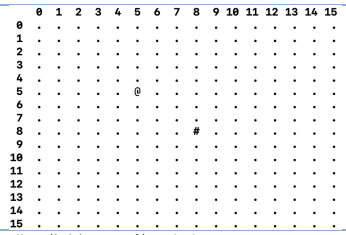
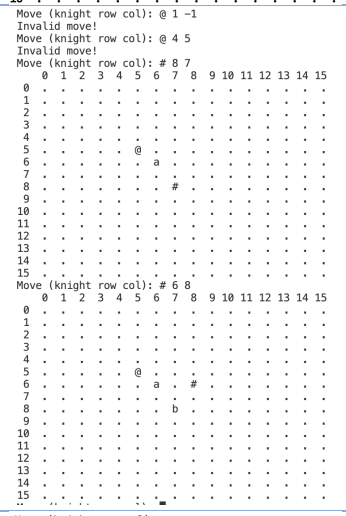
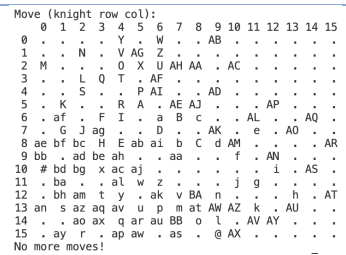
2.6.1 Flow1 board size 6

- This will use our class implement and student's client code

	Inputs	Outputs	Description
1	3 3 3 3	Invalid position(s)!	Two knights with same position, when the board size is 6
2	1 2 5 6	Invalid position(s)!	Out of the board
3	0 1 4 5		initial the board and print the board
5	@ 2 2 #2 4 #4 3 #3 5 @ 3 0 … 21 steps		Can recurrently accept input and invaild input, and output “invalid move” for invaild input and ask users to enter next input.
6	#3 3		When there is no more moves.

2.6.2 Flow2 board size 16

- This will use student's class implement and student's client code

	Inputs	Outputs	Description
1	16 2 5 6	Invalid position(s)!	Two knights with same position, when the board size is 6
2	3 3 3 3	Invalid position(s)!	Out of the board
3	5 5 6 6		initial the board and print the board
4	@ 1 -1 @ 4 5 # 8 7 # 6 8 @ 6 7 … All: 119 steps		Can recurrently accept input and invaild input, and output "invalid move" for invaild input and ask users to enter next input.
5	… # 10 0	 No more moves!	When there is no more moves.