1. Testing

1. Test data for ‘scoreBoard’

	Test Data	Result	Reason to choose Test Data
<1>	Board : InitState Bool : False	0	Test the value when it is ‘InitState’
<2>	Board : (State (1,5) (5,4) [((1,5), P1, 1), ((5,4), P2, 2)]) Bool : True	2	Test whether the ‘chips out’ working as intended, It has to be +1 if it’s true
<3>	Board : (State (1,4) (4,4) [((1,4), P1, 1), ((4,4), P2, 2)]) Bool : False	3	Test when the domino is double

2. Test data for ‘canPlay’

	Test Data	Result	Reason to choose Test Data
<1>	Domino : (3,4) End : R Board : InitState	True	Test when it is ‘InitState’
<2>	Domino : (3,4) End : L Board : (State (4,4) (4,1) [((1,4), P1, 1), ((4,4), P2, 2)])	True	Test the domino with given end

3. Test data for ‘blocked’

	Test Data	Result	Reason to choose Test Data
<1>	Hand : [] Board : (State (4,4) (4,1) [((1,4), P1, 1), ((4,4), P2, 2)])	True	Test when hand is empty
<2>	Hand : [(4,5), (6,6)] Board : InitState	False	Test when board is in ‘InitState’
<3>	Hand : [(3,3), (4.5)] Board : (State (4,4) (4,1) [((1,4), P1, 1), ((4,4), P2, 2)])	False	Test if there is domino that canPlay

4. Test data for ’playDom’

	Test Data	**Result**	**Reason to choose Test Data**
<1>	Player : P1 Domino : (4,5) Board : InitState End : L	Just (State (4,5) (4,5) [((4,5),P1,1)])	Test when board is in ‘InitState’
<2>	Player : P1 Domino : (4,5) Board : (State (4,4) (4,1) [((1,4), P1, 1), ((4,4), P2, 2)]) End : L	Just (State (5,4) (4,1) [((5,4),P1,3), ((4,4),P1,1), ((4,1),P2,2)])	Test when there is a domino to play in the correct given end
<3>	Player : P1 Domino : (4,5) Board : (State (4,4) (4,1) [((1,4), P1, 1), ((4,4), P2, 2)]) End : R	Nothing	Test when there is a domino to play but in the wrong given end
<4>	Player : P1 Domino : (3,5) Board : (State (4,4) (4,1) [((1,4), P1, 1), ((4,4), P2, 2)]) End : R	Nothing	Test when there is no domino to play

5. Test data for ‘possPlays’

	Test Data	**Result**	**Reason to choose Test Data**
<1>	Hand : [] Board : InitState	([],[])	Test when the hand is empty in ‘InitState’
<2>	Hand : [(2,4), (4,5), (5,2), (6,6)] Board : (State (2,3) (3,4) [((2,3), P1, 1), ((3,4), P2, 2)])	([(2,4),(5,2)], [(2,4),(4,5)])	Test valid dominoes for both left and right ends

6. Test data for ‘simplePlayer’

	Test Data	Result	Reason to choose Test Data
<1>	Hand : [(1,2)] Board : InitState Player : P1 Scores : (0,0)	((1,2),L)	Test domino in ‘InitState’
<2>	Hand : [(2,4), (4,5), (5,2), (6,6)] Board : (State (2,3) (3,4) [((2,3), P1, 1), ((3,4), P2, 2)]) Player : P1 Scores : (1,2)	((2,4),L)	To check when simple player has multiple dominoes that can play for both left and right ends

7. Test data for ‘highestScoringDomino’

	Test Data	Result	Reason to choose Test Data
<1>	Hand : [(6,6), (1,2)] Board : InitState Player : P1 Scores : (0,0)	((6,6),L)	Test whether it returns the highest domino among hand in the ‘InitState’
<2>	Hand : [(6,6), (5,3), (4,3)] Board : (State (4,1) (1,6) [((4,1), P1, 1), ((1,6), P2, 2)]) Player : P1 Scores : (1,2)	((3,4),L)	Test if the ‘scoreBoard’ function correctly calculates the score when there are multiple options for domino

8. Test data for ‘blockingPlayer’

	Test Data	Result	Reason to choose Test Data
<1>	Hand : [(5,4), (1,5)] Board : InitState Player : P1 Scores : (0,0)	((5,4),L)	Test whether it returns the highest domino among hand in the ‘InitState’ as intended
<2>	Hand : [(6,3), (6,1), (6,6), (4,5), (6,5)] Board : (State (4,2) (2,6) [((4,2), P1, 1), ((2,6), P2, 2)]) Player : P1 Scores : (2,2)	((5,4),L)	P1 has (6,1), (6,3), (6,5), and (6,6) and P2 has already used (2,6). So, there is a low possibility that P2 has (6,0) or (6,4). It is better to block the left side of the domino board.

9. Test data for ‘particularSpot’

	Test Data	**Result**	**Reason to choose Test Data**
<1>	Hand : [(1,2), (3,4), (5,6)] Board : (State (2,3) (3,0) [((2,3), P1, 1), ((3,0), P2, 2)]) Player : P1 Scores : (1,0)	((1,2),L)	Check game continuity with a valid domino, even when there is no majority in a specific spot
<2>	Hand : [(6,1), (6,5),(6,6)] Board : (State (4,2) (2,6) [((4,2), P1, 1), ((2,6), P2, 2)]) Player : P1 Scores : (2,2)	((6,6),R)	Test that a double domino is prioritised in the majority of a specific spot

10. Test data for ‘smartPlayer’

	Test Data	**Result**	**Reason to choose Test Data**
<1>	Hand : [(4,5), (5,6)] Board : InitState Player : P1 Scores : (0,0)	((4,5), L)	Test if the player returns (4,5) or (5,4) in InitState
<2>	Hand : [(4,5), (5,5), (5,6)] Board : (State (1,1) (1,5) [((1,1),P1,1), ((1,5),P2,2)]) Player : P1 Scores : (0,2)	((5,4), R)	Test the blockingPlayer Strategy works when player has low variety of hand
<3>	Hand : [(1,2), (3,4), (5,6), (4,6)] Board : (State (4,1) (1,0) [((4,1),P1,1), ((1,0),P2,2)]) Player : P1 Scores : (1,0)	((6,4), L)	Test if returns highestScoringDomino strategy when blockingPlayer strategy and particular strategy is not used

11. Match

- 10000 games with an initial hand size of 7, and a target score of 61 with seed of 4
- Test if same players win about 50% of the time
- Test smartPlayer strategies and smartPlayer win simplePlayer
- Testing player against each other

P1 / P2	simplePlayer	highestScoring Domino	blockingPlayer	particularSpot	smartPlayer
simplePlayer	(4995,5005)	X	X	X	X
highestScoring Domino	(208,9792)	(4953,5047)	X	X	X
blockingPlayer	(4467,5533)	(9703,297)	(5032,4968)	X	X
particularSpot	(1403,8597)	(8776,1224)	(1994,8006)	(4986,5014)	X
smartPlayer	(814,9186)	(7953,2047)	(1203,8797)	(3807,6193)	(4972,5028)