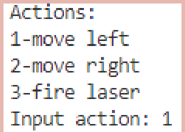
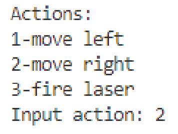
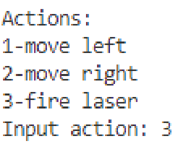
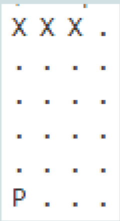
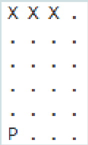
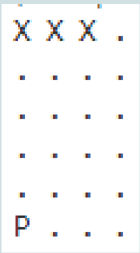
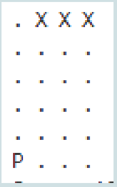
/*12256170 Santos (S19A)
Turn-base_Tiled Shooter_Functions_Test_Cases*/

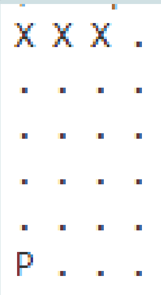
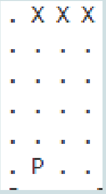
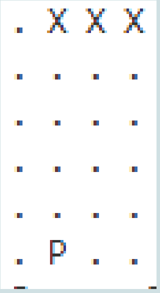
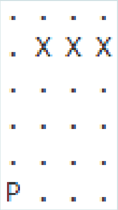
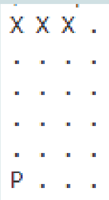
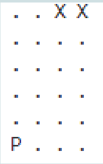
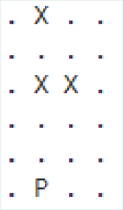
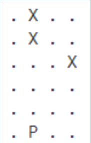
Function	#	Description	Sample Input Data	Expected Output	Actual Output	P/F
getAction	1	Asks the user for an action input	*nAction: 1  Actions: 1-move left 2-move right 3-fire laser Input action: 1	None	None; [Refer to *printfMap().4*]	P
	2	Asks the user for an action input	*nAction: 2  Actions: 1-move left 2-move right 3-fire laser Input action: 2	None	None; [Refer to *printfMap().3*]	P
	3	Asks the user for an action input	*nAction: 3  Actions: 1-move left 2-move right 3-fire laser Input action: 3	None	None; [Refer to *printfMap().2*]	P
moveLeft	1	Moves the player to the left by subtracting 1 to placeP	*nAction: 1 *nPLaceP: 1	*nPLaceP: 0	*nPLaceP:0 ; [Refer to *printfMap().4*]	P
	2	Moves the player to the left by subtracting 1 to placeP	*nAction: 1 *nPLaceP: 2	*nPLaceP: 1	*nPLaceP: 1	P
	3	Moves the player to the leftby subtracting 1 to placeP	*nAction: 1 *nPLaceP: 3	*nPLaceP: 2	*nPLaceP: 2	P
moveRight	1	Moves the player to the right by adding 1 to placeP	*nAction: 2 *nPLaceP	*nPLaceP: 1	*nPLaceP:1 ; [Refer to *printfMap().3*]	P
	2	Moves the player to the right by adding 1 to placeP	*nAction: 2 *nPLaceP: 1	*nPLaceP: 2	*nPLaceP: 2	P
	3	Moves the player to the right by adding 1 to placeP	*nAction: 2 *nPLaceP: 2	nPLaceP: 3	*nPLaceP: 3	P
detectX	1	Detects the enemy that is aligned with the player in its respective column. Here there is only *one* enemy aligned to the player. Calls updateX() function.	*nAction: 3 *nJ1: 0 *nPlaceP: 0	*nI1: 0 *nJ1:0	*nI1: 0 *nJ1:0	P
	2	Detects the enemy that is	*nAction: 3	*nI2:0	*nI2:0	P

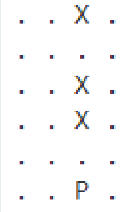
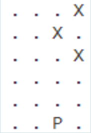
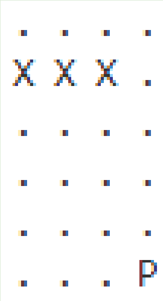
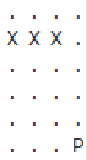
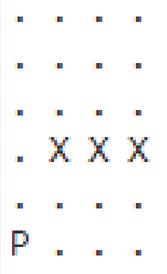
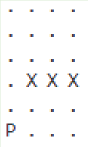
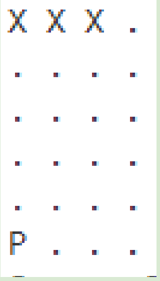
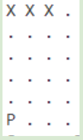
		aligned with the player in its respective column. Here there are only *two* enemies aligned to the player.Calls updateX() function.	*nI2: 2 *nJ2: 1 *nC_2: 4	*nJ2:0 *nC_2: 0	*nJ2:0 *nC_2: 0	
	3	Detects the enemy that is aligned with the player in its respective column. Here there are *three* enemy aligned to the player. Calls updateX() function.	*nAction: 3 *nI1:3 *nJ1:2 *nC_1: 7	*nI1: 0 *nJ1: 2 *nC_1: 0	*nI1: 0 *nJ1: 2 *nC_1: 0	P
updateX	1	One enemy is aligned with the player. The ith position and counter of shooted enemy is set to 0.	*nI:0 *nJ:0 *nCounter: 0 *nScore: 0	*nI:0 *nJ:0 *nCounter: 0 *nScore: 10	*nI:0 *nJ:0 *nCounter: 0 *nScore: 10	P

	2	Two enemies are aligned with the player. The ith position and counter of shooted enemy is set to 0.	*nI:2
			*nJ:1
			*nCounter: 4
			*nScore: 20
				*nI: 0;
				*nJ:0;
				*nCounter: 0;
				*nScore: 30
					*nI: 0;
					*nJ:0;
					*nCounter: 0;
					*nScore: 30
						P

	3	Three enemies are aligned with the player. The ith position and counter of shooted enemy is set to 0.	*nI:3 *nJ:2 *nCounter: 7 *nScore: 70	*nI:0 *nJ:2 *nCounter: 7 *nScore: 80	*nI:0 *nJ:2 *nCounter: 7 *nScore: 08	P
randomX	1	One enemy is aligned with the player.The jth position of enemy is set to 0 and the score is updated by adding 10 pts.	*nJ: 0 *nCounter: 0 *nScore: 0	*nJ: 0 *nScore: 10	*nJ: 0 *nScore: 10	P
	2	Two enemies are aligned with the player.The jth position of enemy is set to 0 and the score is updated by adding 10 pts.	*nJ: 1 *nCounter: 4 *nScore: 20	*nJ: 0 *nScore: 30	*nJ: 0 *nScore: 30	P
	3	Three enemies are aligned with the player.The jth position of enemy is set to 0 and the score is updated by adding 10 pts.	*nJ:2 *nCounter: 7 *nScore: 70	*nJ:2 *nScore: 80	*nJ:2 *nScore: 80	P

moveEnemy	1	The respawned enemy and the rests' $i^{th}$and j^{th} positions are updated depending on their respective count on movement pattern. All counter is 0 thus moving to the right effectively adding 1 to enemy's j^{th} position.	*nI1: 0 *nJ1: 0 *nC_1: 0 *nI2: 0 *nJ2: 1 *nC_2: 0 *nI3: 0 *nJ3: 2 *nC_3: 0	*nI1: 0 *nJ1: 1 *nC_1: 1 *nI2: 0 *nJ2: 2 *nC_2: 1 *nI3: 0 *nJ3: 3 *nC_3: 1	nI1: 0 nJ1: 0 nC_1: 0 nI2: 0 nJ2: 1 nC_2: 0 nI3: 0 nJ3: 2 nC_3: 0	P
	2	The respawned enemy aposition.nd the rests' $i^{th}$and j^{th} positions are updated depending on their respective count on movement pattern. The counters of different enemies vary here.	*nI1: 0 *nJ1: 1 *nC_1: 1 *nI2: 0 *nJ2: 0 *nC_2: 0 *nI3: 0 *nJ3: 2 *nC_3: 4	*nI1: 1 *nJ1: 1 *nC_1: 2 *nI2: 0 *nJ2: 1 *nC_2: 1 *nI3: 2 *nJ3: 3 *nC_3: 5	*nI1: 1 *nJ1: 1 *nC_1: 2 *nI2: 0 *nJ2: 1 *nC_2: 1 *nI3: 2 *nJ3: 3 *nC_3: 5	P
	3	The respawned enemy and the rests' $i^{th}$and j^{th} positions are updated depending on their respective count on movement pattern. The counters of different enemies vary here.	*nI1: 0 *nJ1: 2 *nC_1: 0 *nI2: 0 *nJ2: 2 *nC_2: 1 *nI3: 2 *nJ3: 2 *nC_3: 4	*nI1: 0 *nJ1: 3 *nC_1: 1 *nI2: 1 *nJ2: 2 *nC_2: 2 *nI3: 2 *nJ3: 3 *nC_3: 5	*nI1: 0 *nJ1: 3 *nC_1: 1 *nI2: 1 *nJ2: 2 *nC_2: 2 *nI3: 2 *nJ3: 3 *nC_3: 5	P
printMap	1	Prints the initial game map.  X X X . . . . P . . .	*nI1: 0 *nJ1: 0 *nI2: 0 *nJ2: 1 *nI3: 0 *nJ3: 2 *nPlaceP: 0	nI1: 0 nJ1: 0 nI2: 0 nJ2: 1 nI3: 0 nJ3: 2 *nPlaceP: 0	nI1: 0 nJ1: 0 nI2: 0 nJ2: 1 nI3: 0 nJ3: 2 *nPlaceP: 0  X X X . . . . P . . .	P
	2	Prints the game map following specific sample input data as one of its parameter.  X X X . . . . P . . .	*nAction: 3 *nI1: 0 *nJ1: 0 *nI2: 0 *nJ2: 1 *nI3: 0 *nJ3: 2 *nPlaceP: 0	nI1: 0 nJ1: 1 nI2: 0 nJ2: 2 nI3: 0 nJ3: 3 *nPlaceP: 0	nI1: 0 nJ1: 1 nI2: 0 nJ2: 2 nI3: 0 nJ3: 3 *nPlaceP: 0  . X X X . . . . P . . .	P

	3	Prints the game map following specific sample input data as one of its parameter.  X X X . . . . P . . .	*nAction: 2 *nI1: 0 *nJ1: 0 *nI2: 0 *nJ2: 1 *nI3: 0 *nJ3: 2 *nPlaceP: 0	*nI1: 0 *nJ1: 1 *nI2: 0 *nJ2: 2 *nI3: 0 *nJ3: 3 *nPlaceP: 1	*nI1: 0 *nJ1: 1 *nI2: 0 *nJ2: 2 *nI3: 0 *nJ3: 3 *nPlaceP: 1  . X X X . . . . P . .	P
	4	Prints the game map following specific sample input data as one of its parameters.  . X X X . . . . P . .	*nAction: 1. *nI1: 0 *nJ1: 1 *nI2: 0 *nJ2: 2 *nI3: 0 *nJ3: 3 *nplaceP: 1	*nI1: 1 *nJ1: 1 *nI2: 1 *nJ2: 2 *nI3: 1 *nJ3: 3 *nPlaceP: 0	*nI1: 1 *nJ1: 1 *nI2: 1 *nJ2: 2 *nI3: 1 *nJ3: 3 *nPlaceP: 0  . . . . . X X X . . . . : P . . .	P
	5	Prints the game map following specific sample input data as one of its parameter. Coalition happens.  X X X . . . . P . . .	*nAction: 3 *nI1: 0 *nJ1: 0 *nI2: 0 *nJ2: 1 *nI3: 0 *nJ3: 2	*nI1: 0 *nJ1: 2 *nI2: 0 *nJ2: 2 *nI3: 0 *nJ3: 3	*nI1: 0 *nJ1: 2 *nI2: 0 *nJ2: 2 *nI3: 0 *nJ3: 3  . . X X . . . . P . . .	P
	6	Prints the game map following specific sample input data as one of its parameter. If two enemies and one respawned.  . X . . . . X X . . . . P . .	*nI1: 0 *nJ1: 1 *nI2: 2 *nJ2: 1 *nI3: 2 *nJ3: 2	*nI1: 1 *nJ1: 1 *nI2: 0 *nJ2: 1 *nI3: 2 *nJ3: 3	*nI1: 1 *nJ1: 1 *nI2: 0 *nJ2: 1 *nI3: 2 *nJ3: 3  . X . . . X . . . . . X . . . . P . .	P

		If third enemies and one respawned  . . X . . . . X . . . X . . . . P .	*nI1: 3 *nJ1: 2 *nI2: 0 *nJ2: 2 *nI3: 2 *nJ3: 2	*nI1: 0 *nJ1: 3 *nI2: 1 *nJ2: 2 *nI3: 2 *nJ3: 3	*nI1: 0 *nJ1: 3 *nI2: 1 *nJ2: 2 *nI3: 2 *nJ3: 3  . . . X . . X . . . . X . . . . P .	P
printMap2	1	Prints the game map with invalid nAction input data  . . . . X X X . . . . P	*nAction: 2 *placeP: 3 *nI1: 1 *nJ1: 0 *nI2: 1 *nJ2: 1 *nI3: 1 *nJ3: 2	*nI1: 1 *nJ1: 0 *nI2: 1 *nJ2: 1 *nI3: 1 *nJ3: 2	*nI1: 1 *nJ1: 0 *nI2: 1 *nJ2: 1 *nI3: 1 *nJ3: 2  . . . . X X X . . . . P	P
	2	Prints the game map with invalid nAction input data  . . . . X X X . . . . P . . .	*nAction: 1 *placeP: 0 *nI1: 3 *nJ1: 0 *nI2: 3 *nJ2: 1 *nI3: 3 *nJ3: 2	*nI1: 3 *nJ1: 0 *nI2: 3 *nJ2: 1 *nI3: 3 *nJ3: 2	*nI1: 3 *nJ1: 0 *nI2: 3 *nJ2: 1 *nI3: 3 *nJ3: 2  . . . . X X X . . . . P . . .	P
	3	Prints the game map with invalid nAction input data  X X X . . . . P . . .	*nAction: 1 *placeP: 0 *nI1: 1 *nJ1: 0 *nI2: 1 *nJ2: 1 *nI3: 1 *nJ3: 2	*placeP: 0 *nI1: 1 *nJ1: 0 *nI2: 1 *nJ2: 1 *nI3: 1 *nJ3: 2	*placeP: 0 *nI1: 1 *nJ1: 0 *nI2: 1 *nJ2: 1 *nI3: 1 *nJ3: 2  X X X . . . . P . . .	P