diffEnemyValue()

- Checks if the enemy value for each enemy type are different.
- Compares each witch, wizard, and archers' value attribute to see if they are all different.
- This is important to the implementation of the game because each when each enemy is killed by the player, the player should earn a different money value based on the enemy type.

witchAddBalance()

- Checks if the addBalance function actually updates the players's balance by the witch's value.
- This makes sure that the add balance works for each difficulty by comparing the player balance to the original player's balance + witch's value. If they are the same it works.
- This is important to the implementation of the game because the player's balance should be updated by the value of a witch each time a witch is killed for each difficulty.

wizardAddBalance()

- Checks if the addBalance function actually updates the players's balance by the wizard's value.
- This makes sure that the add balance works for each difficulty by comparing the player balance to the original player's balance + wizard's value. If they are the same it works.
- This is important to the implementation of the game because the player's balance should be updated by the value of a wizard each time a wizard is killed for each difficulty.

archerAddBalance()

- Checks if the addBalance function actually updates the players's balance by the archer's value.
- This makes sure that the add balance works for each difficulty by comparing the player balance to the original player's balance + archer's value. If they are the same it works.
- This is important to the implementation of the game because the player's balance should be updated by the value of an archer each time an archer is killed for each difficulty.

getDistanceTest()

- Checks if the get distance test function actually checks if distance between the place and the enemy is correctly calculated.
- This makes sure that the get distance function works by comparing the actual distance to the distance calculated by the function.
- This is important to the implementation of the game because the distance function is used to determine the distance between the place and the enemy for proximity later on.

inRangeTest()

- Checks if the in range function actually checks if an enemy is in the range or proximity of a place's radius.
- This makes sure that the in range function works by checking if it returns true for a situation where the enemy is in the place's radius.
- This is important to the implementation of the game because the in range function is used to if an enemy is within a place's radius.

outOfRangeTest()

- Checks if the in range function actually checks when an enemy is not in the range or proximity of a place's radius.
- This makes sure that the in range function works by checking if it returns false for a situation where the enemy is not in the place's radius.
- This is important to the implementation of the game because the in range function is used to if an enemy is within a place's radius.

easyEnemyOrder()

- Checks if the deployRightEnemy function actually deploys the enemies in the right order in the right amount for the easy difficulty.
- This makes sure that the proper amount and type of enemies are sent for the easy mode.
- This is important to the implementation of the game because we want to deploy a different number of enemies of different types for the easy difficulty. And when we have waves in M6 it should change as well.

mediumEnemyOrder()

- Checks if the deployRightEnemy function actually deploys the enemies in the right order in the right amount for the medium difficulty.
- This makes sure that the proper amount and type of enemies are sent for the medium mode.
- This is important to the implementation of the game because we want to deploy a different number of enemies of different types for the medium difficulty. And when we have waves in M6 it should change as well.

hardEnemyOrder()

- Checks if the deployRightEnemy function actually deploys the enemies in the right order in the right amount for the hard difficulty.
- This makes sure that the proper amount and type of enemies are sent for the hard mode.
- This is important to the implementation of the game because we want to deploy a different number of enemies of different types for the hard difficulty. And when we have waves in M6 it should change as well.