Requirement B.1: “Speed of car while turning should no larger than 1m/s”

Validation B.1: Run Test B.1

Test B.1: Executes unit tests of speedController node.

```
ros::Subscriber speedTurn_sub = nh.subscribe("/speedTurn", 100,
&CallbackHandler::callback, &mCallbackHandler1);
EXPECT_LE(mCallbackHandler1.result,1);
```

Upon completion, compares value of “/speedTurn” topic we subscribed with 1. The comparison ensures the speed of the car while turning is no larger than 1.

~~~~~~

Requirement B.2: “When the object the car detects is 30+ meters away from it, the linear velocity of the car should no larger than 4”

Validation B.2: Run Test B.2

Test B.2: Executes unit tests of speedController node.

```
ros::Subscriber speedThirtyPlus_sub = nh.subscribe("/speedThirtyPlus", 100,
&CallbackHandler::callback, &mCallbackHandler2);
EXPECT_LE(mCallbackHandler2.result,4);
```

Upon completion, compares value of “/speedThirtyPlus” topic we subscribed with 4. The comparison ensures the speed of the car while turning is no larger than 4.

~~~~~~

Requirement B.3: “When the object the car detects is 20-30 meters away from it, the linear velocity of the car should no larger than 2”

Validation B.3: Run Test B.3

Test B.3: Executes unit tests of speedController node.

```
ros::Subscriber speedTwentyToThirty_sub = nh.subscribe("/speedTwentyToThirty", 100,
&CallbackHandler::callback, &mCallbackHandler3);
EXPECT_LE(mCallbackHandler3.result,2);
```

Upon completion, compares value of “/speedTwentyToThirty” topic we subscribed with 2. The comparison ensures the speed of the car while turning is no larger than 2.

~~~~~~

Requirement B.4: “When the object the car detects is 10-20 meters away from it, the linear velocity of the car should no larger than 1”
~~~~~~

Validation B.4: Run Test B.4

Test B.4: Executes unit tests of speedController node.
ros::Subscriber speedTenToTwenty_sub = nh.subscribe("/speedTenToTwenty", 100, &CallbackHandler::callback, &mCallbackHandler4);
EXPECT_LE(mCallbackHandler4.result,1);

Upon completion, compares value of "/speedTenToTwenty" topic we subscribed with 1. The comparison ensures the speed of the car while turning is no larger than 1.

~~~~~~

Requirement B.5: "The absolute safe distance between the car and the object it detects is 30 meters"

Validation B.5: Run Test B.5

Test B.5: Executes unit tests of speedController node.
ros::Subscriber absSafeDist_sub = nh.subscribe("/absSafeDist", 100, &CallbackHandler::callback, &mCallbackHandler1);
EXPECT_EQ(mCallbackHandler1.result,30);

Upon completion, compares value of "/absSafeDist" topic we subscribed with 30. The absolute safe distance between the car and the object it detects should be equal to 30 meters.

~~~~~~

Requirement B.6: "The quite safe distance between the car and the object it detects is 20 meters"

Validation B.6: Run Test B.6

Test B.5: Executes unit tests of directionController node.
ros::Subscriber quiteSafeDist_sub = nh.subscribe("/quiteSafeDist", 100, &CallbackHandler::callback, &mCallbackHandler2);
EXPECT_EQ(mCallbackHandler2.result,20);

Upon completion, compares value of "/quiteSafeDist" topic we subscribed with 20. The quite safe distance between the car and the object it detects should be equal to 20 meters.

~~~~~~

Requirement B.7: "Distance of vehicle when starting to turn should be exactly 10 meters"

Validation B.7: Run Test B.7
~~~~~~

Test B.7: Executes unit tests of directionController node.
*ros**::**Subscriber turnDist_sub = nh.subscribe("/turnDist", 100, **&**CallbackHandler**::**callback, **&**mCallbackHandler3);*
EXPECT_EQ(mCallbackHandler3.result,10);

Upon completion, compares value of "/turnDist" topic we subscribed with 10. The comparison ensures the distance of the car starting to turn is equal to 10 meters.

~~~~~~

Requirement B.8: "The angle of the car while detected as safe (not within range of an object) is 0 degrees"

Validation B.8: Run Test B.8

Test B.8: Executes unit tests of directionController node.
*ros**::**Subscriber turnAngleSafe_sub = nh.subscribe("/turnAngleSafe", 100, **&**CallbackHandler**::**callback, **&**mCallbackHandler5);*
EXPECT_EQ(mCallbackHandler5.result,0);

Upon completion, compares value of "/turnAngleSafe" topic we subscribed with 0. The comparison ensures the angle of the car while detected as safe is 0 degrees.

~~~~~~

Requirement B.9: "The turning angle of car when turning left should be equal or greater than 3 degrees"

Validation B.9: Run Test B.9

Test B.9: Executes unit tests of directionController node.
*ros**::**Subscriber turnLeftAngleZeroToTen_sub = nh.subscribe("/turnLeftAngleZeroToTen", 100, **&**CallbackHandler**::**callback, **&**mCallbackHandler6);*
EXPECT_GE(mCallbackHandler6.result,3);

Upon completion, compares value of "/turnLeftAngleZeroToTen" topic we subscribed with 3. The comparison ensures the turning angle of car when turning left should be equal or greater than 3 degrees.

~~~~~~

Requirement B.10: "The turning angle of car when turning right should be equal or less than -3 degrees"

Validation B.10: Run Test B.10
~~~~~~

Test B.10: Executes unit tests of directionController node.
ros::Subscriber turnRightAngleZeroToTen_sub = nh.subscribe("/turnRightAngleZeroToTen", 100, &CallbackHandler::callback, &mCallbackHandler7);
EXPECT_LE(mCallbackHandler7.result,-3);

Upon completion, compares value of “/turnRightAngleZeroToTen” topic we subscribed with -3. The comparison ensures the The turning angle of car when turning right should be equal or less than -3 degrees.

~~~~~~

Requirement A.1: “The threshold for the car to decide that there is an exit on right-hand side is equal to 79.9 meters”

Validation A.1: Run Test A.1

Test A.1: Executes unit tests of directionController node.
ros::Subscriber forceTurnRightRange_sub = nh.subscribe("/forceTurnRightRange", 100, &CallbackHandler::callback, &mCallbackHandler1);
EXPECT_EQ(mCallbackHandler1.result,79.9);

Upon completion, compares value of “/forceTurnRightRange” topic we subscribed with 79.9. The comparison ensures the threshold for the car to decide that there is an exit on right-hand side is equal to 79.9 meters.

~~~~~~

Requirement A.2: “The car will decide to turn left if it detects a distance of 79.9 meters or greater on its left side”

Validation A.2: Run test A.2

Test A.2: Executes unit tests of directionController node.
ros::Subscriber forceTurnLeftRange_sub = nh.subscribe("/forceTurnLeftRange", 100, &CallbackHandler::callback, &mCallbackHandler2);
EXPECT_EQ(mCallbackHandler2.result,79.9);

Compares the value of the subscribed topic “/forceTurnLeftRange” with 79.9. This way, we are able to verify that the car detects a distance greater than 79.9 meters when it is in the vicinity of an exit.

~~~~~~

Requirement A3: “The car will turn right only if it detects a distance greater than 79.9 meters meters 40 times in a row”
~~~~~~

Validation A.3: Run test A.3

Test A.3: Executes unit tests of directionController node.

```
ros::Subscriber forceTurnRightCounter_sub = nh.subscribe("/forceTurnRightCounter", 100,
&CallbackHandler::callback, &mCallbackHandler3);
EXPECT_GE(mCallbackHandler3.result,40);
```

Compares the value of the subscribed topic “/forceTurnRightCounter” with 40 to be sure that the system counts 40 iterations of distances greater than 79.9 meters before the car begins its turn.

~~~~~~

Requirement A4: “The car will turn left only if it detects a distance greater than 79.9 meters 40 times in a row”

Validation A.4: Run test A.4

Test A.4: Executes unit tests of directionController node.

```
ros::Subscriber forceTurnLeftCounter_sub = nh.subscribe("/forceTurnLeftCounter", 100,
&CallbackHandler::callback, &mCallbackHandler4);
EXPECT_GE(mCallbackHandler4.result,40);
```

Compares the value of the subscribed topic “/forceTurnLeftCounter” with 40 to be sure that the system counts 40 iterations of distances greater than 79.9 meters before the car begins its turn.

~~~~~~

Requirement A5: “The angle of the wheels when the car begins to turn left is .2 degrees”

Validation A.5: Run test A.5

Test A.5: Executes unit tests of directionController node.

```
ros::Subscriber exitTurningLeftAngle_sub = nh.subscribe("/exitTurningLeftAngle", 100,
&CallbackHandler::callback, &mCallbackHandler8);
EXPECT_EQ(mCallbackHandler8.result,0.2);
```

Compares the value of the subscribed topic “/exitTurningLeftAngle with .2 to be sure that the car turns at an angle of .2 degrees.

~~~~~~

Requirement A6: “The angle of the wheels when the car begins to turn right is -.2 degrees”

Validation A.6: Run test A.6

Test A.6: Executes unit tests of directionController node.
~~~~~~

```
ros::Subscriber exitTurningRightAngle_sub = nh.subscribe("/exitTurningRightAngle", 100,
&CallbackHandler::callback, &mCallbackHandler9);
EXPECT_EQ(mCallbackHandler9.result,-0.2);
```

Compares the value of the subscribed topic “/exitTurningRightAngle with .2 to be sure that the car turns at an angle of .2 degrees.

~~~~~~

Requirement A7: “The car will stop when it is a distance of 55 meters from the starting point.”

Validation A.7: Run test A.7

Test A.7: Executes unit tests of directionController node.

```
ros::Subscriber stopDistance_sub = nh.subscribe("/stopDistance", 100,
&CallbackHandler::callback, &mCallbackHandler11);
EXPECT_EQ(mCallbackHandler11.result,55);
```

Compares the value of the subscribed topic “/stopDistance” to 55 to be sure that the car has stopped 55 meters from its starting point.
~~~~~~