```
/*******************************************************************************
 * Includes
 ******************************************************************************/
#include <gtest/gtest.h>
#include <fstream>
#include <iostream>
#include <string>
#include <vector>
#include <streambuf>

#include "src/Coward.h"
#include "src/LightFactory.h"
#include "src/pose.h"

 /******************************************************
* TEST FEATURE SetUp
*******************************************************/
class CowardBehaviorTest : public ::testing::Test {
public:
  virtual void SetUp() {
    light = factory.Create();
  }
protected:
  csci3081::CowardBehavior cb;
  csci3081::LightFactory factory;
  csci3081::Light * light;
  double speed = 5.0;
};

/*******************************************************************************
 * Test Cases
 ******************************************************************************/

TEST_F(CowardBehaviorTest, NoEntity) {
  std::vector<csci3081::Pose> light_sensors;
  light_sensors.push_back(csci3081::Pose());
  light_sensors.push_back(csci3081::Pose());

  light = NULL;
  csci3081::WheelVelocity wv = cb.CalculateVelocity(light, speed,
     light_sensors);
  csci3081::WheelVelocity expected = csci3081::WheelVelocity(0.0001, 0.0001, speed);

  EXPECT_EQ(wv.left, expected.left) << "FAIL: Default left wheel velocity incorrectly calculted";
  EXPECT_EQ(wv.right, expected.right) << "FAIL: Default right wheel velocity incorrectly calculted";
}

TEST_F(CowardBehaviorTest, CloseDistance) {
  std::vector<csci3081::Pose> light_sensors;
  light_sensors.push_back(csci3081::Pose(200, 190));
  light_sensors.push_back(csci3081::Pose(200, 190));

  csci3081::WheelVelocity wv = cb.CalculateVelocity(light, speed,
     light_sensors);

   double reading_left =  1800.0/std::pow(
     1.08, (light->get_pose()-light_sensors[0]).Length());
   double reading_right = 1800.0/std::pow(
     1.08, (light->get_pose()-light_sensors[1]).Length());

   csci3081::WheelVelocity expected = csci3081::WheelVelocity(reading_left, reading_right, speed);

   EXPECT_EQ(wv.left, expected.left) << "FAIL: Left wheel velocity for close distance incorrectly calculted";
   EXPECT_EQ(wv.right, expected.right) << "FAIL: Right wheel velocity for close distance incorrectly calculted";
}

TEST_F(CowardBehaviorTest, MediumDistance) {
  std::vector<csci3081::Pose> light_sensors;
  light_sensors.push_back(csci3081::Pose(200, 150));
  light_sensors.push_back(csci3081::Pose(200, 150));

  csci3081::WheelVelocity wv = cb.CalculateVelocity(light, speed,
     light_sensors);

   double reading_left =  1800.0/std::pow(
     1.08, (light->get_pose()-light_sensors[0]).Length());
   double reading_right = 1800.0/std::pow(
     1.08, (light->get_pose()-light_sensors[1]).Length());

   csci3081::WheelVelocity expected = csci3081::WheelVelocity(reading_left, reading_right, speed);

   EXPECT_EQ(wv.left, expected.left) << "FAIL: Left wheel velocity for medium distance incorrectly calculted";
   EXPECT_EQ(wv.right, expected.right) << "FAIL: Right wheel velocity for medium distance incorrectly calculted";
```

```
}

TEST_F(CowardBehaviorTest, FarDistance) {
  std::vector<csci3081::Pose> light_sensors;
  light_sensors.push_back(csci3081::Pose(200, 100));
  light_sensors.push_back(csci3081::Pose(200, 100));

  csci3081::WheelVelocity wv = cb.CalculateVelocity(light, speed,
     light_sensors);

   double reading_left =  1800.0/std::pow(
     1.08, (light->get_pose()-light_sensors[0]).Length());
   double reading_right = 1800.0/std::pow(
     1.08, (light->get_pose()-light_sensors[1]).Length());

   csci3081::WheelVelocity expected = csci3081::WheelVelocity(reading_left, reading_right, speed);

   EXPECT_EQ(wv.left, expected.left) << "FAIL: Left wheel velocity for far distance incorrectly calculted";
   EXPECT_EQ(wv.right, expected.right) << "FAIL: Right wheel velocity for far distance incorrectly calculted";
}
```