C++11 code 2.10.14: Unit test for Sub-problem (2-10.i) ➔ GitLab

```
TEST(ProjectionOntoGradients, div_free_test) {
  // Building test mesh
  auto mesh_p = lf::mesh::test_utils::GenerateHybrid2DTestMesh(3);
  // Divergence-free vector field
  const auto f = [](Eigen::Vector2d x) { return Eigen::Vector2d(-x(1), x(0)); };

  // DofHandler for the p.w.  linear Lagrangian finite element
  lf::assemble::UniformFEDofHandler dofh(mesh_p,
                                         {{lf::base::RefEl::kPoint(), 1},
                                          {lf::base::RefEl::kSegment(), 0},
                                          {lf::base::RefEl::kTria(), 0}});
  // Compute solution
  const Eigen::VectorXd sol_vec =
      ProjectionOntoGradients::projectOntoGradients(dofh, f);

  // As stated in the exercise, we would expect the solution to be zero.
  // Hence we check every entry if its (numerically) zero.
  // The GoogleTest framework provides a function we may use:
  /*  EXPECT_NEAR(value 1, value 2, max. difference) */
  const double eps = 1e-15;
  for (std::size_t i = 0; i < sol_vec.size(); ++i) {
    EXPECT_NEAR(sol_vec[i], 0.0, eps);
    // Try testing for equality, you'll see it will fail miserably!
    /* EXPECT_EQ(sol_vec[i], 0.0); */
  }
}
```