C++11 code 3.1.13: Sub-problem (3-1.g): C++ function `approxBoundaryFunctionalValues()` ⇝ GitLab

```cpp
std::vector<std::pair<unsigned int, double>> approxBoundaryFunctionalValues(
    unsigned int L) {
  std::vector<std::pair<unsigned int, double>> result;
  auto meshes = generateTestMeshSequence(L - 1);
  int num_meshes = meshes->NumLevels();
  for (int level = 0; level < num_meshes; ++level) {
    auto mesh_p = meshes->getMesh(level);

    // Set up global FE space; lowest order Lagrangian finite elements
    auto fe_space =
        std::make_shared<lf::uscalfe::FeSpaceLagrangeO1<double>>(mesh_p);

    // Obtain local->global index mapping for current finite element
    space
    const lf::assemble::DofHandler &dofh{fe_space->LocGlobMap()};
    const lf::base::size_type N_dofs(dofh.NumDofs());

    // compute galerkin matrix with alpha = 1.0, gamma = 1.0, beta = 0.0
    auto const_one = [](Eigen::Vector2d x) -> double { return 1.0; };
    auto const_zero = [](Eigen::Vector2d x) -> double { return 0.0; };
    auto A = compGalerkinMatrix(dofh, const_one, const_one, const_zero);

    // compute load vector for f(x) = x.norm()
    auto f = [](Eigen::Vector2d x) -> double { return x.norm(); };
    lf::mesh::utils::MeshFunctionGlobal mf_f{f};
    lf::uscalfe::ScalarLoadElementVectorProvider elvec_builder(fe_space, mf_f);
    Eigen::VectorXd phi(N_dofs);
    phi.setZero();
    AssembleVectorLocally(0, dofh, elvec_builder, phi);

    // solve system of equations
    Eigen::SparseLU<Eigen::SparseMatrix<double>> solver;
    solver.compute(A);
    Eigen::VectorXd u = solver.solve(phi);

    // set up weight function
    auto w = [](Eigen::Vector2d x) -> double { return 1.0; };
    double functional_value = compBoundaryFunctional(dofh, u, w);

    result.push_back({N_dofs, functional_value});
  }
  return result;
}
```