```cpp
#include "/src/Definitions.h"
#include "/src/MeshUtilities.h"
#include "/src/Quadrature.h"
#include "/src/MaterialModelNeoHookean.h"
#include "/src/ElementTestsExtended.h"

#include "ElementTypeTest.h"
#include "ElementTypes.h"
#include "IsoparametricElement.h"

#define SPATIAL_DIMENSION 3
const unsigned int SpatialDimension = SPATIAL_DIMENSION;
const unsigned int DegreesOfFreedom = SpatialDimension;

//Material Model
typedef MaterialModels::NeoHookean<SpatialDimension> MaterialModel;

//Elements and Quadrature Rule
const unsigned int NumberOfQuadPoints = 1;

typedef ElementTypes::Simplex<SpatialDimension>                ElementType;
typedef Elements::IsoparametricElement<MaterialModel,
                                       NumberOfQuadPoints,
                                       ElementType,
                                       DegreesOfFreedom>       Element;
typedef Element::Properties                                    ElementProperties;
typedef Element::Node                                          Node;
typedef Element::Vector                                        Vector;
typedef Element::Point                                         Point;
typedef Element::Stress                                        Stress;
typedef Element::Strain                                        Strain;


int main() {


  //TODO: Define your 1) material model, 2) element type, 3) element properties
  const double mu    = 1.0;
  const double kappa = 2.0;
  MaterialModel materialModel(mu, kappa);
  ElementType elementType;
  ElementProperties elementProperties;
  // ...
  // ...

  // Test ElementType - activate as soon as you created elementType
  //                    (and change the name of the object if you
  //                     did not call it elementType)
  Elements::testElementTypeDerivatives<ElementType>(elementType);

  ignoreUnusedVariables(mu,kappa);

  // TODO: Check out which members quadratureRule has. You should be able to find
  //       all information in /src/Quadrature.h. Yes, for this TODO, you only need
  //       to check out stuff, that's it :)
  const QuadratureRule<SpatialDimension, NumberOfQuadPoints> quadratureRule =
    Quadrature::buildSimplicialQuadrature<SpatialDimension, NumberOfQuadPoints>();

  ignoreUnusedVariables(quadratureRule);

  // TODO: Define some sample points in examplePoints
  array<Vector,SpatialDimension+1> examplePoints;
  examplePoints.fill(Vector::Zero());
  for (unsigned int i = 0; i < (SpatialDimension+1); i++){
    examplePoints[i] = Vector::Random();
  }

  // TODO: Use the above sample points to create sample nodes in exampleNodes
  array<Node,  SpatialDimension+1> exampleNodes;
  for (unsigned int indexNode = 0; indexNode<SpatialDimension+1;indexNode++){
    exampleNodes[indexNode] = Node(indexNode,examplePoints[indexNode]); // ...
  }

```

```
74      //TODO: Create a simplex element
75      Element simplexElement(exampleNodes,
76                             elementProperties,
77                             elementType,
78                             & quadratureRule,
79                             & materialModel);
80
81      //TODO: Finally test the simplex element using the testElementDerivatives
82      //      functionality provided in the namespace Elements as defined in
83      //      ElementTests.h, i.e. Elements::testElementDerivatives
84      Elements::testElementDerivatives<Element>(simplexElement);
85
86
87      // Return
88      return 0;
89    }
90
```