```cpp
#ifndef ELEMENT_EXTERNAL_FORCE
#define ELEMENT_EXTERNAL_FORCE

#include "/src/Definitions.h"

namespace Elements {
namespace ExternalForce {

template <class Element, template<class> class NodeType = NodeWithId>
class ConstantBodyForce {

public:

  static const unsigned int NumberOfNodes    = Element::NumberOfNodes;
  static const unsigned int SpatialDimension = Element::SpatialDimension;
  static const unsigned int DegreesOfFreedom = Element::SpatialDimension;

  typedef Matrix<double, SpatialDimension, 1>          Point;
  typedef NodeType<Point>                              Node;
  typedef Matrix<double, DegreesOfFreedom, 1>          Vector;
  typedef array<Vector, NumberOfNodes>                 NodalDisplacements;
  typedef Matrix<double,
                 NumberOfNodes*DegreesOfFreedom,
                 NumberOfNodes*DegreesOfFreedom>       StiffnessMatrix;
  typedef array<Vector, NumberOfNodes>                 Forces;

  ConstantBodyForce(const Element & element        ,
                    const Vector &  bodyForceVector){

    ignoreUnusedVariables(element);
    ignoreUnusedVariables(bodyForceVector);

    // TODO: Properly define _nodeIds based on information stored in element
    //       as an array with NumberOfNodes entries of type size_t
    // REMINDER: Public members of an object of a class can simply be accessed
    //           via element._somePublicMember

    _nodeIds = element._nodeIds;

    // TODO: Based on the bar's mass, store the mass of each bar
    //       in the array nodalWeights. Note, that all information needed
    //       to obtain a bar's mass can be found in element (such as
    //       element._properties._density or element._X0, element._X1, the
    //       latter two of which can be used to evaluate the length of the
    //       bar, as well as element._properties._area)
    array<double,NumberOfNodes> nodalWeights;
    for (unsigned int nodeIndex = 0; nodeIndex < NumberOfNodes; nodeIndex++) {
      nodalWeights[nodeIndex] = element._properties._density *
      element._properties._area * element._undeformedBarLength / 2;
    }
      //cout<<"nodakWeights is"<<nodalWeights<<endl;

    // TODO: Compute _nodalForces, i.e the equivalent force on each node
    //       resulting from gravitational loading (identical for all nodes)
    //       based on bodyForceVector (g) and the mass (m) stored in
    //       nodalWeights
    //
    // NOTE: _nodalForces is an array with NumberOfNodes entries of type Vector
    for (unsigned int nodeIndex = 0; nodeIndex < NumberOfNodes; nodeIndex++) {
     _nodalForces[nodeIndex] = Vector::Zero();
      _nodalForces[nodeIndex] =  nodalWeights[nodeIndex] * bodyForceVector;
     //cout<<"_nodalForces is"<<_nodalForces[nodeIndex]<< endl;
    }

  }

  double
  computeEnergy(const NodalDisplacements & displacements) const {

    //ignoreUnusedVariables(displacements);

    double energy = 0.;

```

```
73        // TODO: Based on _nodalForces and displacements, evaluate the energy
74        for (unsigned int nodeIndex = 0; nodeIndex < NumberOfNodes; nodeIndex++) {
75          energy -= _nodalForces[nodeIndex] .dot(displacements[nodeIndex]);
76        }
77
78        return energy;
79
80      }
81
82      Forces
83      computeForces(const NodalDisplacements & displacements) const {
84
85        ignoreUnusedVariables(displacements);
86
87        Forces nodalForces;
88
89        // TODO: Evaluate/Set the nodalForces
90        // HINT: Easier than you may think...check your private variables...
91        //       simply be cautious about the sign...
92        for (unsigned int nodeIndex = 0; nodeIndex < NumberOfNodes; nodeIndex++) {
93          nodalForces[nodeIndex] = Vector::Zero();
94          nodalForces[nodeIndex] =- _nodalForces[nodeIndex];
95        }
96
97        return nodalForces;
98
99      }
100
101     // Ignore this function, as you don't need it for now
102     array<size_t, NumberOfNodes>
103     getNodeIds() const {
104       return _nodeIds;
105     }
106
107   private:
108
109     array<Vector, NumberOfNodes> _nodalForces;
110     array<size_t, NumberOfNodes> _nodeIds;
111   };
112
113   }
114   }
115
116   #endif //ELEMENT_EXTERNAL_FORCE
117
```