Project 1

a) $\sigma_{ij} = \frac{\partial W}{\partial \varepsilon_{ij}} = \frac{\partial \left[\frac{\lambda}{2}(\varepsilon_{kk})^2 + \mu \varepsilon_{kl}\varepsilon_{kl}\right]}{\partial \varepsilon_{ij}}$

$= \frac{\lambda}{2}\frac{\partial}{\partial \varepsilon_{ij}}(\varepsilon_{kk})^2 + \mu \frac{\partial(\varepsilon_{kl}\varepsilon_{kl})}{\partial \varepsilon_{ij}}$

$= \frac{\lambda}{2} 2\varepsilon_{kk}\frac{\partial \varepsilon_{kk}}{\partial \varepsilon_{ij}} + \mu\left[\varepsilon_{kl}\frac{\partial \varepsilon_{kl}}{\partial \varepsilon_{ij}} + \varepsilon_{kl}\frac{\partial \varepsilon_{kl}}{\partial \varepsilon_{ij}}\right)$

$= \lambda \varepsilon_{kk}[\delta_{ki}\delta_{kj}] + 2\mu \varepsilon_{kl}\frac{\partial \varepsilon_{kl}}{\partial \varepsilon_{ij}}$

$= \lambda \varepsilon_{kk}\delta_{ij} + 2\mu \varepsilon_{kl}\frac{1}{2}\frac{\partial(\varepsilon_{kl}+\varepsilon_{lk})}{\partial \varepsilon_{ij}}$ as: $\underline{\varepsilon} = \frac{1}{2}(\underline{\varepsilon} + \underline{\varepsilon}^T)$

$= \lambda \varepsilon_{kk}\delta_{ij} + \mu \varepsilon_{kl}[\delta_{ki}\delta_{lj} + \delta_{li}\delta_{kj}]$

$= \lambda \varepsilon_{kk}\delta_{ij} + \mu \varepsilon_{ij} + \mu \varepsilon_{ji}$

$= \lambda \varepsilon_{kk}\delta_{ij} + 2\mu \varepsilon_{ij}$

$\mathbb{C}_{ijkl} = \frac{\partial \sigma_{ij}}{\partial \varepsilon_{kl}} = \frac{\partial}{\partial \varepsilon_{kl}}\left(\lambda \varepsilon_{kk}\delta_{ij} + 2\mu \varepsilon_{ij}\right)$

$= \lambda \frac{\partial(\varepsilon_{kk}\delta_{ij})}{\partial \varepsilon_{kl}} + 2\mu \frac{\partial\left(\frac{1}{2}(\varepsilon_{ij}+\varepsilon_{ji})\right)}{\partial \varepsilon_{kl}}$ as: $\underline{\varepsilon} = \frac{1}{2}(\underline{\varepsilon} + \underline{\varepsilon}^T)$

$= \lambda \delta_{kl}\delta_{ij} + \mu\left(\delta_{ik}\delta_{jl} + \delta_{jk}\delta_{il}\right)$

(b). $\sigma_{ij} = \lambda \varepsilon_{kk} \delta_{ij} + 2\mu \varepsilon_{ij}$ as $\sigma_{ij} = 0$ so $\sigma_{22} = 0$

$\sigma_{22} = \lambda(\varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33})\delta_{22} + 2\mu\varepsilon_{22} = 0$ as: $\varepsilon_{22} = \varepsilon_{33}$

$\lambda\varepsilon_{11} + 2\lambda\varepsilon_{22} + 2\mu\varepsilon_{22} = 0 \Rightarrow \frac{\varepsilon_{22}}{\varepsilon_{11}} = -\frac{\lambda}{2(\lambda+\mu)}$

$\Rightarrow \nu = -\frac{\varepsilon_{22}}{\varepsilon_{11}} = \frac{\lambda}{2(\lambda+\mu)}$

$E = \frac{\sigma_{11}}{\varepsilon_{11}} = \frac{\lambda(\varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33})\delta_{11} + 2\mu\varepsilon_{11}}{\varepsilon_{11}} = \lambda + 2\lambda\frac{\varepsilon_{22}}{\varepsilon_{11}} + 2\mu$

$= \lambda + 2\lambda\left(-\frac{\lambda}{2(\lambda+\mu)}\right) + 2\mu = \lambda + 2\mu - \frac{\lambda^2}{\lambda+\mu}$

$= \frac{\lambda^2 + 2\lambda\mu + 2\lambda\mu + 2\mu^2 - \lambda^2}{\lambda+\mu} = \frac{3\lambda\mu + 2\mu^2}{\lambda+\mu}$

as $\sigma_{11} = \sigma$ $\Rightarrow \sigma_{11} = \lambda(\varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33})\delta_{11} + 2\mu\varepsilon_{11} = \sigma$

$\Rightarrow \lambda(\varepsilon_{11} - \frac{\lambda}{2(\lambda+\mu)} 2\varepsilon_{11}) + 2\mu\varepsilon_{11} = \sigma$

$\Rightarrow \varepsilon_{11} = \frac{\lambda+\mu}{\lambda + 2\lambda\mu + 2\mu^2}\sigma$

$\varepsilon_{22} = -\frac{\lambda}{2(\lambda+\mu)} \cdot \varepsilon_{11} = -\frac{\lambda}{2\lambda + 4\lambda\mu + 4\mu^2}\sigma$

$\varepsilon_{33} = \varepsilon_{22}$

$\sigma_{12} = \lambda(\varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33})\delta_{12} + 2\mu\varepsilon_{12} = 0$

$\Rightarrow \varepsilon_{12} = 0$

in analogously: $\varepsilon_{21} = \varepsilon_{12} = \varepsilon_{13} = \varepsilon_{31} = \varepsilon_{23} = \varepsilon_{32} = 0$

(c) $P_{iJ} = \frac{\partial W}{\partial F_{iJ}}$

$= \frac{\mu}{2}\frac{\partial}{\partial F_{iJ}}(\mathrm{tr}(C_{IK}^{-1}C)) + \frac{\lambda}{2} 2\ln J \frac{\partial \ln J}{\partial F_{iJ}} - \mu\frac{\ln J}{\partial F_{iJ}}$

$= \mu F_{iJ} + \lambda \ln J \frac{1}{J}\frac{\partial J}{\partial F_{iJ}} - \mu\frac{1}{J}\frac{\partial J}{\partial F_{iJ}}$

$= \mu F_{iJ} + \lambda \ln J \frac{1}{J} J F_{Ji}^{-1} - \mu\frac{1}{J} J F_{Ji}^{-1}$

$= \mu F_{iJ} + \lambda \ln J F_{Ji}^{-1} - \mu F_{Ji}^{-1}$

$= \mu_{iJ} + (\lambda \ln J - \mu) F_{Ji}^{-1}$

$$\mathbb{C}_{iJKL} = \frac{\partial P_{iJ}}{\partial F_{KL}}$$

$$= \frac{\partial}{\partial F_{KL}} \left[\mu F_{iJ} + (\lambda \ln J - \mu) F^{-1}_{Ji} \right]$$

$$= \mu \frac{\partial F_{iJ}}{\partial F_{KL}} + \frac{\partial (\lambda \ln J - \mu)}{\partial F_{KL}} F^{-1}_{Ji} + (\lambda \ln J - \mu) \frac{\partial F^{-1}_{Ji}}{\partial F_{KL}}$$

$$= \mu \frac{\partial F_{iJ}}{\partial F_{KL}} + \lambda \frac{\partial \ln J}{\partial F_{KL}} F^{-1}_{Ji} + (\lambda \ln J - \mu) \frac{\partial F^{-1}_{Ji}}{\partial F_{KL}}$$

$$= \mu \delta_{iK} \delta_{JL} + \lambda \frac{1}{J} \frac{\partial J}{\partial F_{KL}} F^{-1}_{Ji} + (\lambda \ln J - \mu)(-F^{-1}_{JK} F^{-1}_{Li})$$

$$= \mu \delta_{iK} \delta_{JL} + \frac{\lambda}{J} J F^{-1}_{LK} F^{-1}_{Ji} - (\lambda \ln J - \mu)(F^{-1}_{JK} F^{-1}_{Li})$$

$$= \mu \delta_{iK} \delta_{JL} + \lambda F^{-1}_{LK} F^{-1}_{Ji} - (\lambda \ln J - \mu)(F^{-1}_{JK} F^{-1}_{Li})$$

(d). for undeformed ground state $\underline{F} = \underline{I}$. thus $\ln J = 0$

thus

$$\mathbb{C}_{iJKL} = \mu \delta_{iK} \delta_{JL} + \lambda F^{-1}_{LK} F^{-1}_{Ji} - (\lambda \ln J - \mu)(F^{-1}_{JK} F^{-1}_{Li})$$

$$= \mu \delta_{iK} \delta_{JL} + \lambda \delta_{LK} \delta_{Ji} + \mu \delta_{JK} \delta_{Li}$$

$$= \lambda \delta_{KL} \delta_{iJ} + \mu (\delta_{iK} \delta_{JL} + \delta_{JK} \delta_{iL})$$

It agrees well with the linear elastic model:

$$\mathbb{C}_{ijKL} = \lambda \delta_{KL} \delta_{ij} + \mu (\delta_{iK} \delta_{jL} + \delta_{jK} \delta_{iL})$$

```cpp
#include "Definitions.h"
#include "PostProcessorVtk.h"


int main() {

  // Geometrical properties
  // TODO: Set positive integer for numberOfElementsX and numberOfElementsY
  int numberOfElementsX = 10;
  int numberOfElementsY = 10;

  cout << "Let's build a mesh with numberOfElementsX=" << numberOfElementsX << " and numberOfElementsY=" << numberOfElementsY << endl;
  cout << "This should give a grant total of " << numberOfElementsX*numberOfElementsY*2 << " elements" << endl;

  // TODO: Set positive side lengths
  double sideLengthX = 10.0;
  double sideLengthY = 10.0;

  cout << "The side lengths are: sideLengthX=" << sideLengthX << ", sideLengthY=" << sideLengthY << endl;

  // Initialization of mesh
  Mesh rectangularTriangleMesh;

  // Incremental side length of one element
  // TODO: based on numberOfElementsX,numberOfElementsY
  //       and sideLengthX,sideLengthY, define dx,dy
  double dx = sideLengthX / numberOfElementsX;
  double dy = sideLengthY / numberOfElementsY;

  cout << "The incremental side lengths are: dx=" << dx << ", dy=" << dy << endl;




  // TODO: Add all nodes into the public member _nodes of rectangularTriangleMesh.
  //       You may want to use a for-loop or even a nested-for-loop to achieve this.
  // HELP: This is how you create three nodes and it onto the aforementioned vector
  //       (Reminder: 'nested' means a for-loop inside a for-loop).
  Vector2d nodeLocation;
  for (double y = 0.0; y <= sideLengthY; y += dy) {
    for (double x = 0.0; x <= sideLengthX; x +=dx){
        nodeLocation(0) = x;
        nodeLocation(1) = y;
        Node node(int((numberOfElementsX+1)*(y/dy)+ x/dx), nodeLocation);
        rectangularTriangleMesh._nodes.push_back(node);
    }
  }




  // TODO: The goal of the next section is to create a triangular element as shown in the
  //       assignment sheet and to add it onto the _connectivity public member variable of
  //       rectangularTriangleMesh.
  //       In order to build the entire mesh, this then has to be repeated for all elements, the total
  //       number of which is determined via numberOfElementsX and numberOfElementsY. You may want to use a
  //       nested for-loop
  // HINT: See your job similar to the one of a floor tiler, but instead of explicitely saying where every
  //       single tile goes, you tell the program how to place one or two tiles and then let it repeat this
  //       process numberOfElementsX times in the x-direction and again repeat this numberOfElementsY times
  //       in the y-direction.
```

```
64
65      // HELP: This is how we create ONE triangular element:
66    // elementNumber = numberOfElementsX*numberOfElementsY*2
67      for (double i = 0.0; i < sideLengthX; i +=dx){
68          for (double j = 0.0; j < sideLengthY; j+=dy){
69              array<int, 3> connection1;  // this holds the node IDs of the three nodes
                of the element
70              array<int, 3> connection2;
71              connection1[0] = int ((numberOfElementsX+1)*(j/dy)+i/dx);
72              connection1[1] = int ((numberOfElementsX+1)*(j/dy)+i/dx+1);
73              connection1[2] = int ((numberOfElementsX+1)*(j/dy+1)+i/dx+1);
74              connection2[0] = int ((numberOfElementsX+1)*(j/dy)+i/dx);
75              connection2[1] = int ((numberOfElementsX+1)*(j/dy+1)+i/dx+1);
76              connection2[2] = int ((numberOfElementsX+1)*(j/dy+1)+i/dx);
77              // HELP: This is how you add it onto the _connectivity member of
                rectangularTriangleMesh
78              rectangularTriangleMesh._connectivity.push_back(connection1);
79              rectangularTriangleMesh._connectivity.push_back(connection2);
80          }
81
82      }
83      // Make VTK file
84      PostProcessors::Vtk::makeUndeformedMeshFile(rectangularTriangleMesh,string("Mesh"));
85
86      cout << "Succesfully ran main and created .vtu file" << endl;
87
88      return 0;
89    }
90
91
92
93
```