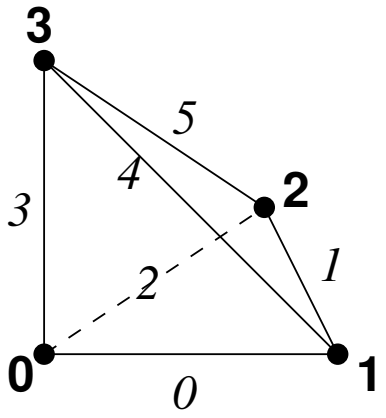

Programming in C^{++}:
Let the vertices and edges of the reference tet be numbered according to the figure below.



Recall the barycentric coordinates λ_α for $\alpha = 0, ..., 3$, where

$$\lambda_0(\boldsymbol{\xi}) = 1 - \xi_1 - \xi_2 - \xi_3,$$
$$\lambda_i(\boldsymbol{\xi}) = \xi_i, \quad i = 1, ..., 3.$$

Recall the Nédélec shape functions $\boldsymbol{\varphi}_{\alpha\beta} = \lambda_\alpha \nabla\lambda_\beta - \lambda_\beta \nabla\lambda_\alpha$ with

$$\mathbf{curl}\, \boldsymbol{\varphi}_{\alpha\beta} = 2\,(\nabla\lambda_\alpha \times \nabla\lambda_\beta).$$

To model small vectors from $\mathbb{R}^n$ and matrices from $\mathbb{R}^{n\times m}$ with $m, n \in \{2, 3\}$, it is recommended to use `vec.hh` and `mat.hh` (downloadable from the website, together with a demo `matvecdemo.cc`).

22 23 Complete and implement the following class modelling the affine linear transformation from the reference tet $\widehat{T}$ to the physical tet T (given by the coordinates of the four vertices).

```
class ElTrans{
public:
   ElTrans(const Vec<3>& p0, const Vec<3>& p1,
           const Vec<3>& p2, const Vec<3>& p3);
   void transform (const Vec<3>& xi, Vec<3>& x);
   void getJacobian(Mat<3, 3>& F);
   void getInvJacobian(Mat<3, 3>& Finv);
   double getJacobiDet();
   ...
};
```

The method `transform` should transform reference coordinates `xi`$= \xi$ to real coordinates `x`$= x_0 + F\,\xi$. The method `getJacobian` should return the Jacobi matrix F of the transformation, `getInvJacobian` should return the inverse Jacobi matrix F^{-1} and `getJacobiDet` should return the Jacobi determinant.

24 Complete and implement the following class modelling the Nédélec element:

```
class NedelecTet{
public:
   NedelecTet(const Vec<4, int>& vnums);
   void calcShape (int e, const Vec<3>& xi, Vec<3>& shape);
   void calcCurlShape (int e, const Vec<3>& xi, Vec<3>& curlShape);
   ...
};
```

Here, `vnums` contains the global indices of the vertices. The edges should be oriented with respect to `vnums`. The numbering of the edges (and the corresponding shape functions) is according to the figure above. In `calcShape`, `shape`=$\boldsymbol{\varphi}_e(\xi)$. In `calcCurlShape`, `curlShape`=$\mathbf{curl}\,(\boldsymbol{\varphi}_e(\xi))$.