Programming in C^{++}
(prepare your code on a USB stick or send it by e-mail until 30min before the tutorial)

In this tutorial we consider Courant's finite element. The reference triangle is given by

$$\Delta = \{\xi \in \mathbb{R}^2 : \xi_1 \geq 0,\ \xi_2 \geq 0,\ \xi_1 + \xi_2 \leq 1\},$$

with vertices $\xi^{(0)} = (0,0)$, $\xi^{(1)} = (1,0)$, and $\xi^{(2)} = (0,1)$, the space of shape functions is P_1, and the nodal variables are the evaluations at the three vertices. Recall that the nodal shape functions are given by

$$\begin{aligned} p^{(0)}(\xi) &= 1 - \xi_1 - \xi_2, \\ p^{(1)}(\xi) &= \xi_1, \\ p^{(2)}(\xi) &= \xi_2. \end{aligned}$$

To model *small* vectors from $\mathbb{R}^n$ and $n \times m$ matrices, where $m,\, n \in \{2, 3\}$, I recommend to use `vec.hh` and `mat.hh` (downloadable from the website, together with a demo `matvecdemo.cc`). There 0-based indices are used throughout, for example:

$\xi \in \mathbb{R}^2 \ \leftrightarrow$	`Vec<2> xi`	$\xi_1 \ \leftrightarrow$	`xi[0]`
		$\xi_2 \ \leftrightarrow$	`xi[1]`

To model the *type* of a scalar function depending on a vector in $\mathbb{R}^2$ use

```
typedef double (*ScalarField)(const Vec<2>& x);
```

$\boxed{25}$ Write two functions

```
double calcShape (int i, const Vec<2>& xi);
Vec<2> calcDShape (int i, const Vec<2>& xi);
```

that compute the *value* $p^{(\alpha)}(\xi)$ and the *gradient* $\nabla_\xi\, p^{(\alpha)}(\xi)$ of a nodal shape function, respectively, where `xi`$=\xi$ and `i`$=\alpha$.

$\boxed{26}$ Complete and implement the following class modelling the affine linear transformation x_δ from Δ to an *arbitrary* non-degenerate triangle δ:

$$x \;=\; x_\delta(\xi) \;=\; x_0 + J\,\xi,$$

where x_0 is the image of $(0,0)$.

```
class ElTrans {
public:
  ElTrans(const Vec<2>& x0, const Vec<2>& x1, const Vec<2>& x2);
  void transform (const Vec<2>& xi, Vec<2>& x);
  void getJacobian (Mat<2, 2>& J);
  ...
};
```

Above, `x0`, `x1`, `x2` are the three vertices of δ. The method `transform` should transform reference coordinates `xi`$=\xi$ to real coordinates `x`$=x_\delta(\xi)$. The method `getJacobian` should return the Jacobi matrix J of the transformation.

$\boxed{27}$ Add two more methods to `class ElTrans`:

```
double jacobiDet ();
void getInvJacobian (Mat<2, 2>& invJ);
```

The first should return the Jacobi determinant $\det J$, the second one should return `invJ`$=J^{-1}$.

$\boxed{28}$ Write a function

```
void calcLaplaceElMat (const Vec<2>& x0, const Vec<2>& x1,
                       const Vec<2>& x2, Mat<3, 3>& elMat);
```

that computes the element stiffness matrix `elMat`$=K_r$ associated to an element δ_r (given by the three vertices `x0`, `x1`, and `x2`), i. e.

$$(K_r)_{\alpha\beta} = \int_{\delta_r} \nabla_x p^{(r,\alpha)}(x) \cdot \nabla_x p^{(r,\beta)}(x)\, dx = \int_\Delta \big(J_r^{-T} \nabla_\xi p^{(\alpha)}(\xi)\big) \cdot \big(J_r^{-T} \nabla_\xi p^{(\beta)}(\xi)\big) \det(J_r)\, d\xi.$$

Hint: Consider only the above formula on the reference element. Use `calcDShape` to get $\nabla_\xi p^{(\alpha)}(\xi)$, and `ElTrans` to get $\det J$ and J_r^{-1}. Note finally that J_r^{-T} and $\nabla_\xi p^{(\alpha)}$ are constant on Δ.

$\boxed{29}$ Write a function

```
void calcSourceElVec (const Vec<2>& x0, const Vec<2>& x1,
    const Vec<2>& x2, ScalarField f, Vec<3>& elVec);
```

that approximates the element load vector f_r given by

$$(f_r)_\alpha = \int_{\delta_r} f(x)\, p^{(r,\alpha)}(x)\, dx = \int_\Delta f(x_{\delta_r}(\xi))\, p^{(\alpha)}(\xi) \det(J_r)\, d\xi,$$

using the following quadrature rule on Δ:

$$\int_\Delta g(\xi)\, d\xi \approx \frac{1}{6}\Big[g(\tfrac{1}{6}, \tfrac{1}{6}) + g(\tfrac{4}{6}, \tfrac{1}{6}) + g(\tfrac{1}{6}, \tfrac{4}{6})\Big].$$

Show that this quadrature rule is exact for $g \in P_2$.

Hint: Proceed similarly as in exercise $\boxed{29}$ and use `ElTrans` to get $x_{\delta_r}(\xi)$. Note that ξ must *loop* over the three integration points.

$\boxed{30}$ Write a function

```
void calcMassElMat (const Vec<2>& x0, const Vec<2>& x1,
                    const Vec<2>& x2, Mat<3, 3>& elMat);
```

that computes the element mass matrix M_r given by

$$(M_r)_{\alpha\beta} = \int_{\delta_r} p^{(r,\alpha)}(x)\, p^{(r,\beta)}(x)\, dx$$

Hint: Transform to the reference element as done in the previous two exercises.

Test all your functions, i. e. apply them to concrete parameters and output the results! At minimum use $f(x, y) = 1$ and test $\delta_r = \Delta$ as well as the triangle with the vertices $(1, 1)$, $(1.5, 1)$, and $(1.25, 1.5)$.