Let $\Omega = (0,1)$, $\Gamma_D = \{0\}$, and $\Gamma_R = \{1\}$. Consider the following one-dimensional boundary value problem: Find $u(x)$ such that

$$\begin{aligned} -u''(x) &= f(x) && \text{for } x \in \Omega\,, \\ u(x) &= g_D(x) && \text{for } x \in \Gamma_D\,, \\ u'(x) &= \alpha(x)\big(g_R(x) - u(x)\big) && \text{for } x \in \Gamma_R\,. \end{aligned} \tag{3.1}$$

We discretize this problem using the FEM with Courant elements. Consider the nodes $0 = x_0 < x_1 < \cdots < x_{n_h} = 1$ which define a mesh (subdivision) $\mathcal{T}_h$ of Ω with the elements $T_k = (x_{k-1},\, x_k)$, $k = 1, \ldots, n_h$. We introduce the finite element space

$$V^h := \{v_h \in C(\overline{\Omega}) : v_{h|T} \in P_1 \text{ for all } T \in \mathcal{T}_h\}$$

whose basis is given by the nodal (hat) functions φ_i, $i = 0, \ldots, n_h$, defined by

$$\varphi_i(x_j) = \delta_{ij} \quad \text{for } i,\, j = 0, \ldots, n_h\,.$$

In the following exercises we start to develop a C/C^{++}program that will allow us to compute the finite element approximation u_h of the weak solution u to (3.1).

Notation: We indicate input parameters of a C/C^{++}function by "↓" and output parameters by "↑". If not pointed out explicitly, the functions discussed below have no return value.

13 Write a function `ElementStiffnessMatrix(↓xa, ↓xb, ↑element_matrix)` which for given nodes `xa` $= x_{k-1}$ and `xb` $= x_k$ returns the element stiffness matrix `element_matrix` $= K_h^{(k)}$ on the element T_k, defined by

$$K_h^{(k)} = \begin{bmatrix} \displaystyle\int_{T_k} \big(\varphi'_{k-1}(x)\big)^2\,dx & \displaystyle\int_{T_k} \varphi'_{k-1}(x)\,\varphi'_k(x)\,dx \\ \displaystyle\int_{T_k} \varphi'_k(x)\,\varphi'_{k-1}(x)\,dx & \displaystyle\int_{T_k} \big(\varphi'_k(x)\big)^2\,dx \end{bmatrix} \quad \text{for } k = 1, \ldots, n_h\,.$$

Hint: You can use the type `typedef double Mat22[2][2];` to represent a two-by-two matrix.

14 Write a function `ElementLoadVector(↓(*f)(x), ↓xa, ↓xb, ↑element_vector)` which for a given function `f` $= f \in C[0,1]$ and the nodes `xa` $= x_{k-1}$ and `xb` $= x_k$ returns the 2-dimensional element load vector `element_vector` $= f_h^{(k)}$ on the element T_k, defined by

$$f_h^{(k)} = \begin{pmatrix} \displaystyle\int_{T_k} f(x)\,\varphi_{k-1}(x)\,dx \\ \displaystyle\int_{T_k} f(x)\,\varphi_k(x)\,dx \end{pmatrix} \quad \text{for } k = 1, \ldots, n_h\,.$$

Use the trapezoidal rule to approximate above integrals:

$$\int_a^b g(x)\,dx \;\simeq\; \frac{b-a}{2}\big[g(a) + g(b)\big]\,.$$

Hint: You can use the following types and function header:

```
typedef double (*RealFunction)(double x);
typedef double Vec2[2];
void ElementLoadVector (RealFunction f, double xa, double xb, Vec2& element_vector);
```

15 Define a data type `Mesh` which contains all the information on the mesh $\mathcal{T}_h$, see also your lecture notes.

Hint: Use `class` in C^{++}, or `struct` in C.

16 Define an efficient data type `Matrix` for the sparse stiffness matrix K_h exploiting the fact that K_h is tridiagonal.

Hint: Use `class` or `struct`.

Consider now the case $\Gamma_D = \emptyset$, $\Gamma_R = \{0, 1\}$, and $\alpha(x) = 0$, which corresponds to the homogeneous Neumann boundary conditions.

17 Write a function `AssembleStiffnessMatrix(↓mesh, ↑matrix)` that assembles the global $(n_h+1) \times (n_h+1)$ stiffness matrix `matrix` $= K_h$ for a given subdivision `mesh` $= \mathcal{T}_h$ of Ω.

Hint: Set $K_h = 0$, then start with $K_h^{(1)}$ and loop over all elements T_k to update the matrix K_h. On each element T_k, use the function `ElementStiffnessMatrix` to compute $K_h^{(k)}$ and pay attention to put the entries of $K_h^{(k)}$ at the correct positions in the global matrix K_h.

18 Write a function `AssembleLoadVector(↓(*f)(x), ↓mesh, ↑vector)` that assembles the global (n_h+1)-dimensional load vector `vector` $= \underline{f}_h$ for a given mesh `mesh` $= \mathcal{T}_h$ of Ω.

Hint: Set $\underline{f}_h = 0$, then start with $\underline{f}_h^{(1)}$ and loop over all elements T_k to update the vector $\underline{f}_h$. On each element T_k, use the function `ElementLoadVector` to compute $\underline{f}_h^{(k)}$ and pay attention to add the entries in the right place.

Test the implemented data types and functions using some simple examples, e. g., consider equidistant nodes x_i for different values of n_h, and simple functions $f(x) = 1$, $f(x) = x$, etc.

Provide your solution on a USB stick or send it by e-mail before Monday 9.45am.