UCLouvain EPL-INMA 18 juin 2025	**Exam : LINMA2710** Scientific Computing	First name : Last name : NOMA :

Important instructions:

- The exam duration is 2 hours.
- Please start your answer just below each question (when possible) and continue on a new sheet of paper if necessary. Put your name on all sheets.
- As mentioned in the course outline, only personal handwritten notes and annotated printouts of the material posted on the course Moodle page are allowed. All the rest, including electronic devices (laptop, smartphone, etc.), is forbidden.
- Do not use red ink.
- The size of the answer box is chosen with a quite conservative margin so do not worry if your answer is shorter than the box.

UCLouvain EPL-INMA 18 juin 2025	**Exam : LINMA2710** Scientific Computing	First name : Last name : NOMA :

1. Fill the gap between lines 3 and 21 in this C++ code. The code is meant to print "4.4", which is the area of a room of length 2.2 and breadth 2.0.

```
1
2  #include <iostream>
3  using namespace std;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21 int main() {
22
23     Room room1;
24
25     room1.length = 2.2;
26     room1.breadth = 2.0;
27
28     cout << "Area of Room =  " << room1.calculate_area() << endl;
29
30     return 0;
31 }
```

2. Write a finite-volume scheme with only three (distinct) vertices for solving in C++ the heat equation on a disk

$$\Omega = \{x \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq R^2\}$$

with Dirichlet boundary conditions using a finite volume method. You can choose where to put the vertices; at least one should be on the boundary of Ω and one in the interior of Ω.

Comments: For this question, there is no length limitation, but concise answers (i.e., short and clear) are preferred. There is not a unique adequate way to answer the question, but the answer should at least contain: (i) a paragraph that relates to PDEs, (ii) a paragraph that introduces a numerical method, and (iii) a C++ code. Since you do not have access to a Voronoi library, feel free to restrict the vertices to a simple configuration. The C++ code should produce an array `U` that contains the numerical solution. In the C++ code, the problem description should be hard coded. Hence do not address input/output aspects.

3. Below is the implementation of the multiplication of a sparse matrix in Column-Sparse Compressed (CSC) format with a dense vector. This implementation uses OpenMP to parallelize the multiplication.

   Here is example explaining the CSC format. Consider the following 3×4 sparse matrix:

$$A = \begin{bmatrix} 0 & 4 & 0 & 8 \\ 2 & 0 & 0 & 0 \\ 0 & 5 & 0 & 6 \end{bmatrix}$$

   The CSC (Compressed Sparse Column) representation is:

   - `nzval` = [2, 4, 5, 8, 6]
   - `rowval` = [1, 0, 2, 0, 2]
   - `colptr` = [0, 1, 3, 3, 5]

   Explanation:

   - `nzval` stores the nonzero values, column by column.
   - `rowval` stores the row index for each value in `nzval`.
   - `colptr` indicates the starting index in `nzval` for each column (with the last entry pointing one past the end).

```
private:
    std::vector<float> nzval;
    std::vector<size_t> rowval;
    std::vector<size_t> colptr;
    size_t m;
    size_t n;
public:
    std::vector<float> multiply(const std::vector<float>& x) const {
        if (x.size() != n)
            throw std::invalid_argument("Size mismatch");
        std::vector<float> y(m, 0);
        #pragma omp parallel for
        for (size_t j = 0; j < n; ++j) {
            for (size_t idx = colptr[j]; idx < colptr[j + 1]; ++idx) {
                size_t i = rowval[idx];
                float val = nzval[idx];
                y[i] += val * x[j];
            }
        }
        return y;
    }
```

   When trying your function, you sometime get 1) incorrect results and 2) degraded performance when using multiple threads.

(a) What is the mistake in the `multiply` function shown above causing these issues? Explain why it is a problem, especially in the context of parallel execution with OpenMP.

(b) How can you fix the mistake? No need to write code, you can explain in words but then be precise.

4. You now want to distribute the matrix multiplication over multiple compute nodes of a cluster. The sparse matrix is large and should be distributed over the compute nodes.

   Then, the matrix-vector product will be computed for **many** different vectors. So we want connect the compute nodes so as to minimize the cost of the matrix-vector product, not the cost of distributing the matrix.

   You have access to d compute nodes and enough material to connect d pairs of compute nodes together. Using these connections, the unidirectional communication of k bytes takes a time equal to $\alpha + \beta k$. Let γ be the time taken by a compute node to execute one arithmetic operation and z be the number of nonzero entries in the whole sparse matrix.

   (a) How would you interconnect the compute nodes to minimize the time of the matrix-vector product ? Which part of the matrix would be stored in each compute node?

(b) Assuming the interconnection topology you choose in the first part is used. Suppose a dense vector is loaded in the memory of the first compute node. Which MPI collectives would you use to compute the product of that vector with the sparse matrix and retrieve the result on the first compute node? What would be the time complexity of this operation in terms of $\alpha, \beta, \gamma, d, m, n$ and z (you may not need all these 7 variables in your answer).