General rules

- You have 2 hours to complete the test. When applicable, people with special facilities have 2h20 minutes in total.
- The exam is "closed book", meaning that you can only make use of the material given to you.
- You are supposed to write the codes in the Python programming language, but syntactic errors are allowed as far as the written algorithm can be well understood.
- **Do not use while statements.**
- **Do not use global variables in functions.**
- Keep the names of the variables and functions as stated in the question.
- **If you do not follow these instructions you will not receive any points in the respective question.**

Identities

$$\sum_{i=1}^{n} i = n\frac{n+1}{2}, \qquad \sum_{i=1}^{n} i^2 = \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6}, \qquad \sum_{i=1}^{n} i^3 = \frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4}$$

Questions

1. Given a natural number n, the following algorithm can be defined:

   - In case that n is even, then redefine n by halving it.
   - In case that n is odd, then redefine n by tripling it and then adding 1 to the result.

   Lothar Collatz conjectured that this algorithm would eventually reach 1 no matter the starting value of n. Write a program which verifies this conjecture for all starting values $n < 100$. If the algorithm does not terminate in 1000 steps, you may consider to have found a counter example to Collatz's conjecture, for the purpose of the question.

   The algorithm is assumed to have converged if 1 is reached within a tolerance of 10^{-15}.

   The program should print for every n if the algorithm converged within 1000 steps or not, and the number of steps if convergence was achieved.

2. A prime number is a natural number that is only divisible by itself and 1. Write a python function `primes(n)` that receives n and returns a list with the first n prime numbers, just by iterating through all natural numbers and verifying that it is a prime. Note that 1 is not a prime number. *In this question, a `while` loop is allowed!*

3. Let $A, B \in \mathbb{R}^{n\times n}$ be two lower triangular matrices.

   For all questions, compute the complexity of the algorithm in terms of Big-O notation $O(a \cdot n^p)$, indicating the values for $a > 0$ and p.

   Make use of the sparsity patterns of A and B in all questions, so that you obtain the lowest possible complexity, i.e., smallest possible a and p.

   Do not use `numpy` functions, except `numpy.zeros`, `numpy.shape`. Slicing is allowed.

   Do not use conditional statements within the algorithms.

   (a) Write a python function `lowtri_matvec(A, b)` that receives A and a vector $\vec{b} \in \mathbb{R}^n$ as arguments and returns the matrix-vector product $A\vec{b}$.

   (b) Write a python function `lowtri_matmat(A, B)` that given A and B, returns AB. How does the complexity, in terms of both a and p, compare with the multiplication of two non-sparse matrices?