Vectors and Matrices

We will create here set of classes (or a single class maybe?) capable of performing operations between matrices and vectors that can be used in multiple tasks such as rotation of objects in multidimensional spaces (handy for computer graphics) and matrix inversion (useful for data fitting, graphical manipulations, AI engines and countless other subjects)

Create a set of classes, exploiting the clear inheritance relating the elements, that implements

a) columns vectors ***C*** (intended as N-rows x 1 column collection of doubles)
b) row vectors ***R*** (intended as 1-row x N column collection of doubles)
c) 2 dimensional square matrices ***M*** (intended as N-rows x N columns collection of doubles)
d) (EXTRA POINTS) Non square matrices having a different number of columns and rows

Implement now the following operations, properly overloading the relevant operators between two vectors of the same species (two column or two row vectors) or two square matrices of the same dimension

e) **V**+**V** or ***C***+***C*** or ***M***+***M***
f) **V**-**V** or ***C***-***C*** or ***M***-***M***
g) a****V***, a****C*** or a****M*** (where a is a double)

implement the following ***TRANSPOSE*** (ref: https://en.wikipedia.org/wiki/Transpose) operation mapping a *column* vector into a *row* vector (and vice versa) or exchanging all the rows and columns into a square matrix. Verify that the function returns the proper class type and think about what happens when you apply the function TWICE as TRANSPOSE(TRANSPOSE(X))….

h) TRANSPOSE: $(\mathbf{X})^{-T}$=**Y** (where **X** can be a ***C***, ***V***, or ***M*** and ***Y*** will consequently be a ***V***, ***C***, or a ***M)***
i) (EXTRA POINTS) implement the above functions using polymorphism AND in a computational efficient form (possibly reconsider your class implementation…).

implement now the following operations between vectors of different or the same groups (here ***C*** represents an N-dimensional *column* vector, ***R*** represents a N-dimensional *row* vector, overloading, for the scalar product, the proper operator

j) *Scalar product* ***C*V***=r or ***V*******C***=s ***C*******C'***=r' or ***V*******V'***=s' (where r,s,r',s' are just doubles, ref: https://en.wikipedia.org/wiki/Dot_product) [1]
k) *Cross product* (between vectors of the same kind) CROSS(***C,C'***)=***C***" or CROSS(***V,V')***=***V"*** (ref: https://en.wikipedia.org/wiki/Cross_product)

Implement now, overloading the proper operator, the following key operation between matrix (NxN) and column vectors (Nx1) or between matrixes or the same dimension (ref: https://en.wikipedia.org/wiki/Matrix_multiplication#Square_matrix_and_column_vector)

l) ***M*******C=C1*** (where C1 is a new vector)
m) ***M*******M1***=***M2*** (matrix product of two different matrices)

[1] scalar product (a * V = r)
matrix product (M * M = MM)
dot product (or inner product) (C · V = r)
crossproduct (or outer product) (C × V = r)

n) (EXTRA POINT) implement in an efficient way (think transpose) the following ***V*M=V1***

Now that all the preliminary work is done and the machinery is constructed, let's do something useful.

We will implement an algorithm to solve a N equation linear system that we write in the form

$$\{a_{11}x + a_{12}y + a_{13}z = Q_1 \ a_{21}x + a_{22}y + a_{23}z = Q_2 \ a_{31}x + a_{32}y + a_{33}z = Q_3$$

Where the "a_{ij}" and "Q_j" are doubles (real numbers) and x,y,z are our unknown.

This, as you will realize from the implementation you have done above, can be written as

M*V=Q

Where

$M = a_{11} \ a_{12} \ a_{13} \ a_{21} \ a_{22} \ a_{23} \ a_{31} \ a_{32} \ a_{33}$, $V = x \ y \ z$ and $Q = Q_1 \ Q_2 \ Q_3$

While the example is for a 3 equation case, the implementation will cover a more generic N equation case

o) Implement, by using the Gaussian Elimination Method (ref: https://en.wikipedia.org/wiki/Gaussian_elimination) an algorithm to solve the generic form of the ***M*V=Q*** equation where ***M*** is a NxN matrix and ***Q*** is a Nx1 column vector

p) (EXTRA POINT) what happens when, during the elimination process, a matrix row completely filled with zero is reached? (check "singular matrix" meaning)

(EXTRA POINT)

q) Solve the following equation ***M*K=I*** where, this time, ***M,K*** and ***I*** are *all NxN matrices and **I** is the identity matrix .*

r) Indicate now how, the so obtained ***K*** matrix can be used to solve directly any system of the form _***M*V=Q*** *as long as the matrix M does not change but for any choice of the vector Q*