Assignment 1: Vectors and Matrices

This assignment is to get you familiar with programming in a high-level symbolic language.

Your objectives, apart from getting to use OCaml, include

- being able to build a set of functions around a particular representation,
- being able to reason about the correctness of the representation
- defining tests to check and demonstrate that your implementations work as expected
- documenting the code you write, explaining why "your logic" is correct
- analysing the efficiency of your programs, and examining the trade-offs, and possibilities for improvements given a usage context
- defining tests to show that your programs run efficiently.

The task is to model the data types of (a) ***vectors*** and (b) *m x n* ***matrices.***

The basic types and type constructions you will use are OCaml **floats** and OCaml **lists**.

Vectors:

Define vectors as lists of floats. Write efficient and documented programs to perform the following operations on vectors:

1. *vdim*: vector -> int, returns the dimension of a given vector
2. *mkzerov*: int -> vector, given a dimension *n > 0,* returns the zero vector of that dimension
3. *isvzerov*: vector -> bool, checks if a given vector is a zero vector
4. *addv*: vector -> vector -> vector, adds two vectors *v1* and *v2* (of the same dimension)
5. *scalarmultv*: float -> vector -> vector, given a scalar *c* and a vector *v*, performs the scalar multiplication
6. *dotprodv:* vector -> vector -> float, given two vectors *v1* and *v2* of the same dimension, returns their dot product *v1 . v2*
7. *crossprodv:* vector -> vector -> vector, given two vectors *v1* and *v2* in 3 dimensions, returns their cross product *v1 x v2.* (In general, for extra credit, you can define a function *crossprodv* of *n-1* vectors with dimension *n).*

Matrices*:*

Define matrices as lists of lists of floats (in row major form). Write efficient and documented programs to perform the following operations on vectors:

1. *mdim*: matrix -> int * int, returns the dimensions of a given matrix
2. *mkzerom*: int -> int -> matrix, given a dimension *m, n > 0*, returns the zero *m x n* matrix
3. *iszerom*: matrix -> bool, checks if a given matrix is a zero matrix

4. *mkunitm*: int -> matrix, given a dimension *m > 0*, returns the unit *m x m* (square) matrix
5. *isunitm*: matrix -> bool, checks if a given matrix is a unit (square) matrix
6. *addm*: matrix -> matrix -> matrix, adds two matrices m*1* and *m2* (of the same dimensions)
7. *scalarmultm*: float -> matrix -> matrix, given a scalar *c* and a matrix *m*, performs the scalar multiplication
8. *multm*: matrix -> matrix -> matrix, multiply two matrices *m1* and *m2* (assuming their dimensions allow them to be multiplied
9. *transm: matrix -> matrix,* transpose a given matrix
10. *detm*: matrix -> float, compute the determinant of a matrix (assuming it is a square matrix).
11. *invm*: matrix -> matrix, return the inverse of a given matrix (if defined).

    Define suitable exceptions (the TAs will give you the standard names and type to use, so that your programs can run on their test data)

    If you can think of other meaningful functions, do add them to your set of functions.