C for Science - Practical Exercise #3

1. Write three C functions with the following prototypes:

```
double v3dot (double *, double *);
void v3cross (double *, double *, double * );
void v3crosscross (double *, double *, double *, double *);
```

   where,

   (a) `v3dot(a,b)` calculates the scalar product of the two vectors **a** and **b**:

   $$\texttt{v3dot(a,b)} = \mathbf{a} \bullet \mathbf{b} = \texttt{a[0]b[0] + a[1]b[1] + a[2]b[2]}$$

   (b) `v3cross(a, b, r)` computes the vector product of **a** and **b** and stores the result in **r**:

   $$\mathbf{r} = \mathbf{a} \times \mathbf{b} = \begin{pmatrix} \texttt{a[1]b[2]} - \texttt{a[2]b[1]} \\ \texttt{a[2]b[0]} - \texttt{a[0]b[2]} \\ \texttt{a[0]b[1]} - \texttt{a[1]b[0]} \end{pmatrix}$$

   (c) `v3crosscross(a, b, c, r)` computes the triple vector cross product of **a**, **b** and **c** storing the result in **r**.

   $$\mathbf{r} = \mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = (\mathbf{a} \bullet \mathbf{c})\mathbf{b} - (\mathbf{a} \bullet \mathbf{b})\mathbf{c}$$

2. Write a `main` function that:

   (a) prompts for:

      i. the 3 components of vector **a**,

      ii. the 3 components of vector **b**, and

      iii. the 3 components of vector **c**.

   (b) computes, and prints to screen, the scalar and vector products of **a** and **b**, and the triple vector cross product $\mathbf{a} \times (\mathbf{b} \times \mathbf{c})$.

3. Test the code on the two data sets:

   (a) $\mathbf{a} = (1, 1, 0)^{\mathrm{T}}, \quad \mathbf{b} = (0, 1, 1)^{\mathrm{T}}, \quad \mathbf{c} = (1, 0, 1)^{\mathrm{T}}$, and

   (b) $\mathbf{a} = (1, -1, 2)^{\mathrm{T}}, \quad \mathbf{b} = (2, 1, 1)^{\mathrm{T}}, \quad \mathbf{c} = (1, 2, 11)^{\mathrm{T}}$

4. For data set (a) above, print $\mathbf{a} \times (\mathbf{b} \times \mathbf{c})$ and $(\mathbf{a} \times \mathbf{b}) \times \mathbf{c}$ to text files `vector1.txt` and `vector2.txt` respectively. Are they the same?

5. The Identity Matrix, **I** is the $n \times n$ square matrix, with a leading diagonal of ones and zeros elsewhere:

   $$I_1 = \begin{bmatrix} 1 \end{bmatrix}, \; I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \; I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \; \cdots, \; I_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

   (a) Create functions to print and free matrices (you can copy these from today's lecture).

   (b) Create a function to create I_n with the following prototype:

```
double ** matrixI(int n);
```

   Similar to today's matrix-allocating function, this should allocate and return an $n \times n$ matrix (of type `double **`) from the heap. The returned matrix should have the appropriate values set for I_n.

   (c) Create a `main` function to prompt for n and print I_n to `stdout`.