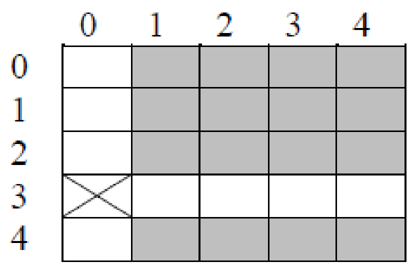
Problem:

Write a method ***double det(double matrix[][])*** that takes as its argument a 2D array representing a square matrix and returns its determinant recursively computed as follows:

- Determinant of a 1-by-1 matrix is its only element;
- Determinant of a larger matrix ***A*** of size ***int n det(A)*** is the sum over all rows of the expressions ***A[k][0] * det(B) * (-1)***^{k}, where ***B*** is a smaller matrix called a minor and made up of all elements of ***A*** not belonging to the row ***k*** and column ***0*** (***k = 0, 1, 2, … , n-1***). For example, if ***n = 5*** and ***k = 3***, then ***B*** consists of the shaded elements



Hint:

In other words, expresses the determinant as a weighted sum, where each element of the chosen row is multiplied by the determinant of a smaller matrix(*minor*) formed by deleting the row and column corresponding to that element, plus an additional cofactor which alternates between -1 and 1.

For example:

|1 2 3| |5 6| |2 3| |2 3|
|4 5 6| = 1*|8 9| - 4*|8 9| + 7*|5 6|
|7 8 9|

= 1*(5*|9| - 8*|6|) – 4*(2*|9| - 8*|3|) +7*(2*|6| - 5*|3|) = -3 + 24 - 21 = 0

The base case is just that the determinant of a single matrix is the value of its single element. The method above implements the additional case when n=2. This is just a special case of the above formula and is probably intended as an optimization.

A *minor* of an element in a matrix is the determinant of the matrix that is formed by removing the elements in the same row and column of the element. The minor of a 4 x 4 matrix is the determinant of a 3 x 3 matrix.

P.S.

I would not recommend using this method to evaluate determinants unless the matrices are all very small because it's extremely slow for large matrices - given an n*n matrix, it takes O(n!) time. This comes from the fact that each method call must evaluate n determinants of size (n-1), so the total number of recursive calls is n*(n-1)*(n-2)*...*3*2*1.