When you have completed the lab, show this sheet and any associated programs to your lab instructor, who will record that you have completed the lab. If you do not finish this exercise during the lab, show the instructor what you have done at the end of the lab and be sure to complete it in the next few days. Also, after showing your work to the instructor and before leaving the lab, please *delete any files* you have created on the Desktop or in any other folder.

If you have any questions, **ask** your lab instructor or a consultant immediately! They are in the lab to help you learn the material.

1 A bit more DNA analysis

Write a function **findPattern(dna,pat)** to return a vector of the locations where the string **pat** appears in **dna**. As before, the strings contain only the characters 'A', 'C', 'G', 'T'. Assume that vector **dna** is longer than or equal in length to **pat**.

2 Determinant of a 3×3 matrix

Write a function **myDeterminant(x)**, where **x** is a 3×3 matrix. Use the following formula:

$$\det\left(\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}\right) = a \det\left(\begin{pmatrix} e & f \\ h & i \end{pmatrix}\right) - b \det\left(\begin{pmatrix} d & f \\ g & i \end{pmatrix}\right) + c \det\left(\begin{pmatrix} d & e \\ g & h \end{pmatrix}\right)$$

You may use the built-in function **det** to find the determinants of 2×2 matrices and, after you complete your function, test your code.

Vectorized code

MATLAB allows one to write vectorized code, or code that perform basic operations on multiple cells at the same time (with one statement). Consider code that we have written in the past involving vectors:

```
% Given vectors a, b of length 5
for k= 1:5
   c(k)= a(k) + b(k);
end
```

The above code *uses* vectors, but is not *vectorized code*, since the operations (addition, assignment) are performed on *individual cells*, one at a time. Another way to say this is that the *operands* are scalars (just one cell of a vector/matrix). Vectorized code perform operations (arithmetic, relational, logical, and assignment) on multiple cells in one statement:

```
% Given vectors a, b of length 5
c= a + b;
```

Notice that in the above vectorized code the operands are vectors. You can use vectorized code as long as the operands "line up," i.e., are vectors/matrices of the same dimensions. For multiplication, division, and exponentiation, you need to use a dot (.) in front of the operator. Examples:

```
% Given vectors a, b of length 5
d= a .* b;  % Result:  d(k) is a(k)*b(k)  for all k
f= a .^ b;  % Result:  f(k) is a(k)^b(k)  for all k
g= a == b;  % Result:  For all k,  g(k) is 1 if a(k)==b(k), otherwise g(k) is 0
```

What about using built-in functions? When you use a function that operates on a matrix, you do not know whether the code in that function are vectorized or not. We *do not* consider function calls to be vectorized code. For example, the statement `x= sum(v)` where `v` is a vector is *not* vectorized code—it just passes a vector to the function `sum`.

3 Lots of extra work without vectorized code!

Consider the 9/29 lecture example `minInNeighborhood.m`. (Download it from the course website.) We used vectorized code to build a "border" of values around the original matrix `M`.

1. Modify **minInNeighborhood** to *not* use vectorized code (in constructing the border). This means creating a matrix of the correct size and then assigning values into the matrix one cell at a time.
2. Further modify **minInNeighborhood** to *not* use function **minInMatrix** or MATLAB's built-in function **min**. This makes **minInNeighborhood** more difficult to write but it's good practice on loops and matrices. (It also shows why it's nice to write and use user-defined functions!)

4 Review exercises from last week's section

Be sure that you complete/review last week's section exercises. Ask if you have questions—don't just look at the solutions.

Delete the your files from the computer before you leave!