Homework 10

This is a nice short homework that illustrates a neat trick that can be done by having a lvalue returned by a function. You need to write 3 lines of code, one of which is "}" and one of which is given in "What you need to do" below.

Background:

Multidimensional arrays are a problem with C++. We can do things like

double x[j][k][l][m];

but j, k and l need to be constants in most implementations of C++. We can do something like

double* x = new double[j][k][l][m];

and have all of j, k, l, and m not be constants, but now indexing x[p][q][r][s] requires expressions like

(x + p*k*l*m + q*l*m + r*m + s) (1)

Remember, however, that if we have an lvalue on the right had side of an assignment, C++ will convert it to a rvalue. For example, if we have x = y, where y is a reference to an object, C++ will use the value at the location specified by y to assign to x. As well, if we have the assignment y = x; with the same x and y as before, C++ will use the lvalue of y to specify the memory location where the rvalue specified by x will be stored.

It turns out we can solve the first problem with indexing by using the property of lvalues and rvalues in the above paragraph, an using operator overloading. The overloaded ( ) operator in C++ can take variable numbers of arguments, so we can replace x[p][q][r][s] with x(p,q,r,s) and declare an operator( ) as double& operator( )(int p, int q, int r, ints) and return the location found by the subscript expression in (1) above as the address of the double being accessed, which the reference double& will be.

What you need to do:

In the Matrix class, create an overloaded double& operator( )(int i, int j); for a two dimensional matrix. The expression for the element returned for an M x N array x is (x + i*N + j). Use it with the included main.cpp to both retrieve and store values into the function.

Grading points:

3 points for the desired output

7 points for implementing the operator( ) correctly.