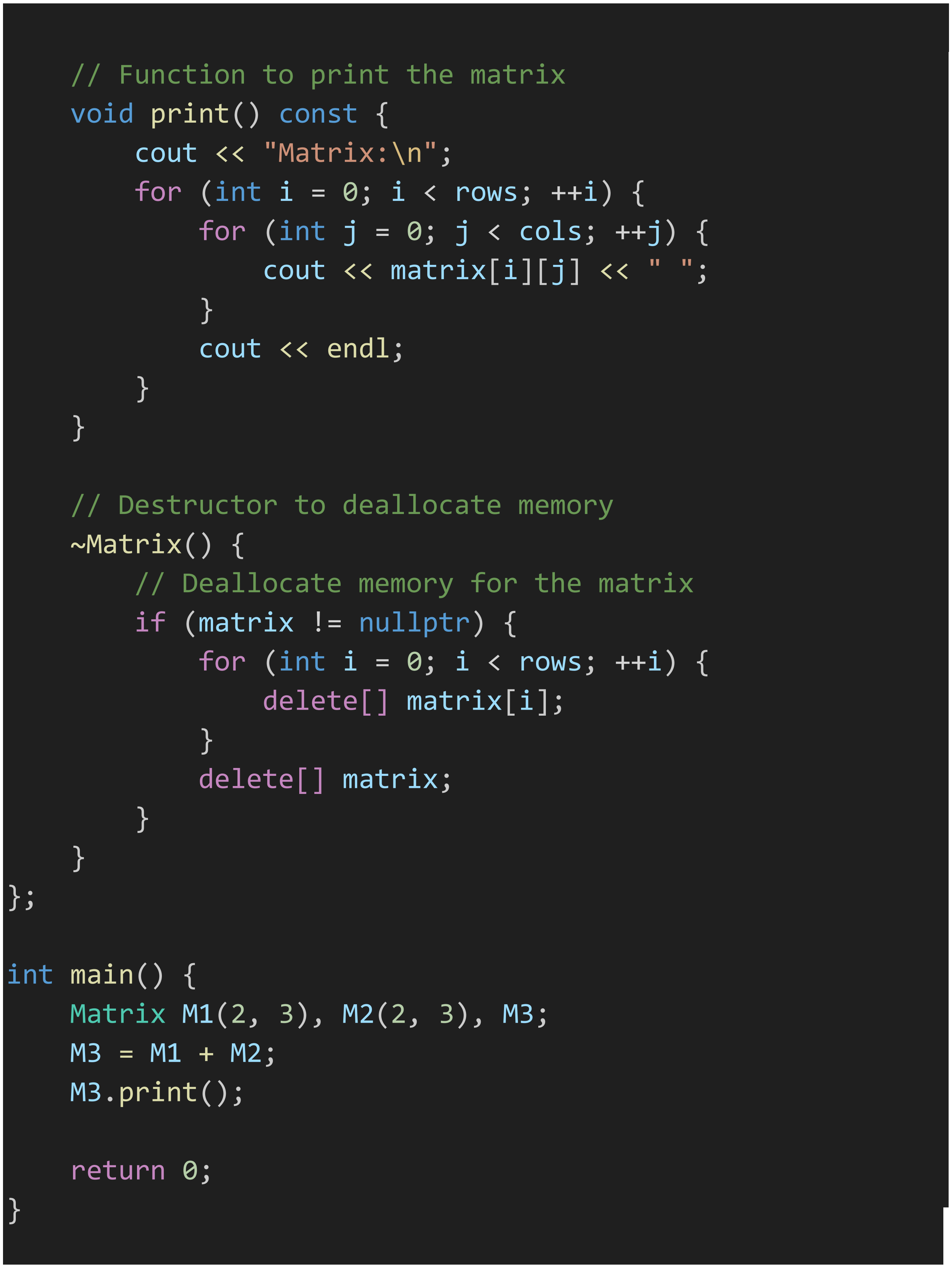
Matrix Operator Addition Overloading

Q5

```
#include <iostream>

using namespace std;

class Matrix {
    int rows, cols;
    int** matrix;

public:
    // Default constructor
    Matrix() {
        rows=0;
        cols=0;
        matrix=nullptr;}

    // Parameterized constructor to initialize the matrix
with given rows and columns
    Matrix(int r, int c)
    {
        rows=r;
    cols=c;
        // Dynamically allocate memory for the matrix
        matrix = new int*[rows];
        for (int i = 0; i < rows; ++i) {
            matrix[i] = new int[cols];
        }

        // Input values from the user
        cout << "Enter matrix elements:\n";
        for (int i = 0; i < rows; ++i) {
            for (int j = 0; j < cols; ++j) {

                cin >> matrix[i][j];
            }
        }
    }



    // Overloaded assignment operator to assign one matrix
to another
    Matrix& operator=(const Matrix& other) {
        if (this != &other) { // Check for self-assignment
            // Deallocate memory for the current matrix
            if (matrix != nullptr) {
                for (int i = 0; i < rows; ++i) {
                    delete[] matrix[i];
                }
                delete[] matrix;
            }
            // Copy dimensions
            rows = other.rows;
            cols = other.cols;
            // Dynamically allocate memory for the new
matrix
            matrix = new int*[rows];
            for (int i = 0; i < rows; ++i) {
                matrix[i] = new int[cols];
                // Copy elements from the other matrix
                for (int j = 0; j < cols; ++j) {
                    matrix[i][j] = other.matrix[i][j];
                }
            }
        }
        return *this;
    }

    // Overloaded addition operator to add two matrices
    Matrix operator+(const Matrix& x) const {
        // Check if dimensions match
        if (rows != x.rows || cols != x.cols) {
            cerr << "Error: Matrix dimensions
mismatch!\n";
            exit(1);
        }

        // Create a new matrix to store the result
        Matrix result(rows, cols);

        // Add corresponding elements
        for (int i = 0; i < rows; ++i) {
            for (int j = 0; j < cols; ++j) {
                result.matrix[i][j] = matrix[i][j] +
x.matrix[i][j];
            }
        }

        return result;
    }
```



```cpp
    // Function to print the matrix
    void print() const {
        cout << "Matrix:\n";
        for (int i = 0; i < rows; ++i) {
            for (int j = 0; j < cols; ++j) {
                cout << matrix[i][j] << " ";
            }
            cout << endl;
        }
    }

    // Destructor to deallocate memory
    ~Matrix() {
        // Deallocate memory for the matrix
        if (matrix != nullptr) {
            for (int i = 0; i < rows; ++i) {
                delete[] matrix[i];
            }
            delete[] matrix;
        }
    }
};

int main() {
    Matrix M1(2, 3), M2(2, 3), M3;
    M3 = M1 + M2;
    M3.print();

    return 0;
}
```