Matrix operator overloading Equal and assignment operator

```
#include<iostream>
using namespace std;

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

public:
    // Constructor
    Matrix(int r, int c) : rows(r), cols(c) {
        mat = new int* [rows];
        for (int i = 0; i < rows; ++i)
            mat[i] = new int[cols];
    }

    // Destructor
    ~Matrix() {
        for (int i = 0; i < rows; ++i)
            delete[] mat[i];
        delete[] mat;
    }

    // Overloading * operator
    Matrix operator*(Matrix& m);

    // Overloading << operator
    friend ostream& operator<<(ostream& out, Matrix& m);

    // Overloading >> operator
    friend istream& operator>>(istream& in, Matrix& m);

    // Overloading assignment operator
    Matrix& operator=(const Matrix& m);
```

```
    // Overloading equality operator
    bool operator==(const Matrix& m) ;
};

Matrix& Matrix::operator=(const Matrix& m) {
    if (this != &m){

    for (int i = 0; i < rows; ++i){
        for (int j = 0; j < cols; ++j){
            mat[i][j] = m.mat[i][j];
            }
        }
    }
    return *this;
}

bool Matrix::operator==(const Matrix& m) {


    for (int i = 0; i < rows; ++i){
        for (int j = 0; j < cols; ++j){
            if (mat[i][j] != m.mat[i][j]){
                return false;
            }
        }
    }

    return true;
}

Matrix Matrix::operator*(Matrix& m) {
    if (cols != m.rows) {
        cout << "Matrix multiplication not possible." << endl;
        exit(0);
    }
```

```
    Matrix result(rows, m.cols);
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < m.cols; j++) {
            result.mat[i][j] = 0;
            for (int k = 0; k < cols; k++) {
                result.mat[i][j] += this->mat[i][k] *
m.mat[k][j];
            }
        }
    }
    return result;
}

ostream& operator<<(ostream& out, Matrix& m) {
    for (int i = 0; i < m.rows; i++) {
        for (int j = 0; j < m.cols; j++) {
            out << m.mat[i][j] << " ";
        }
        out << endl;
    }
    return out;
}

istream& operator>>(istream& in, Matrix& m) {
    cout << "Enter elements: ";
    for (int i = 0; i < m.rows; i++) {
        for (int j = 0; j < m.cols; j++) {
            in >> m.mat[i][j];
        }
    }
    return in;
}

int main() {
    int rows1, cols1, rows2, cols2;
```

```
    cout << "Enter number of rows and columns for first matrix:
";
    cin >> rows1 >> cols1;
    cout << "Enter number of rows and columns for second
matrix: ";
    cin >> rows2 >> cols2;

    if (cols1 != rows2) {
        cout << "Matrix multiplication not possible." << endl;
        return 0;
    }

    Matrix m1(rows1, cols1), m2(rows2, cols2);

    cout << "For matrix 1" << endl;
    cin >> m1;
    cout << "For matrix 2" << endl;
    cin >> m2;

    // Check if m1 is equal to m2
    if (m1 == m2)
        cout << "m1 is equal to m2" << endl;
    else
        cout << "m1 is not equal to m2" << endl;

    // Create and assign m3 after m1 and m2 have been filled
    Matrix m3 = m1 * m2;

    cout << "Resultant Matrix: " << endl;
    cout << m3;

    return 0;
}
```