```cpp

//Sartaj Khan Problem Set 1 Part 1
#include "Matrix3x3.h"
#include <cassert>
#include <vector>

Matrix3x3 Matrix3x3::operator*(const Matrix3x3& aOther) const noexcept {
    Vector3D aVector1 = *this * static_cast<Vector3D>(aOther.column(0));
    Vector3D aVector2 = *this * static_cast<Vector3D>(aOther.column(1));
    Vector3D aVector3 = *this * static_cast<Vector3D>(aOther.column(2));
    Matrix3x3 aMatrix = Matrix3x3(aVector1, aVector2, aVector3);
    return aMatrix.transpose();
}

float Matrix3x3::det() const noexcept {
    return (row(0).x() * (row(1).y() * row(2).w() - row(1).w() * row(2).y()) - row(0).y() * (row(1).x() * row(2).w() - row(1).w() * row(2).x()) + row(0).w() * (row(1).x() * row(2).y() - row(1).y() * row(2).x()));
}

bool Matrix3x3::hasInverse() const noexcept {
    return (this->det() != 0);
}

Matrix3x3 Matrix3x3::transpose() const noexcept {
    return Matrix3x3(column(0), column(1), column(2));
}

Matrix3x3 Matrix3x3::inverse() const {
    assert(this->hasInverse());
    std::vector<Vector3D> invRows;
    for (int i = 0; i < 3; i++) {
        std::vector<float> aVector;
        for (int j = 0; j < 3; j++) {
            float aInvElement = ((row((j + 1) % 3)[(i + 1) % 3] * row((j + 2) % 3)[(i + 2) % 3] - row((j + 1) % 3)[(i + 2) % 3] * row((j + 2) % 3)[(i + 1) % 3]) / det());
            aVector.push_back(aInvElement);
        }
        invRows.push_back(Vector3D(aVector[0], aVector[1], aVector[2]));
    }
    return Matrix3x3(invRows[0], invRows[1], invRows[2]);
}

std::ostream& operator<<(std::ostream& aOStream, const Matrix3x3& aMatrix) {
    return aOStream << "[" << aMatrix.row(0) << "," << aMatrix.row(1) << "," << aMatrix.row(2) << "]";
}

```