```cpp
//  Matrix3x3_PS1.cpp
//  problem_Sets
//  Created by H M Asfaq Ahmed Shihab on 23/3/2024.
#define _USE_MATH_DEFINES
#include "Matrix3x3.hpp"
#include <cassert>
#include <cmath>
#include <iostream>

using namespace std;

// Multiplication of matrix
Matrix3x3 Matrix3x3::operator*(const Matrix3x3& aOther) const noexcept {
    Vector3D lRow1(row(0).dot(aOther.column(0)),
     row(0).dot(aOther.column(1)), row(0).dot(aOther.column(2)));
    Vector3D lRow2(row(1).dot(aOther.column(0)),
     row(1).dot(aOther.column(1)), row(1).dot(aOther.column(2)));
    Vector3D lRow3(row(2).dot(aOther.column(0)),
     row(2).dot(aOther.column(1)), row(2).dot(aOther.column(2)));
    return Matrix3x3(lRow1, lRow2, lRow3);
}

// Determinant of the matrix
float Matrix3x3::det() const noexcept {
    const Vector3D& lRow1 = row(0);
    const Vector3D& lRow2 = row(1);
    const Vector3D& lRow3 = row(2);
    return lRow1.x() * (lRow2.y() * lRow3.w() - lRow2.w() * lRow3.y()) -
        lRow1.y() * (lRow2.x() * lRow3.w() - lRow2.w() * lRow3.x()) +
        lRow1.w() * (lRow2.x() * lRow3.y() - lRow2.y() * lRow3.x());
}

// Transpose of the matrix
Matrix3x3 Matrix3x3::transpose() const noexcept {
    return Matrix3x3(column(0), column(1), column(2));
}

// Invertibility
bool Matrix3x3::hasInverse() const noexcept {
    return det() != 0;
}

// Inverse matrix
Matrix3x3 Matrix3x3::inverse() const noexcept {
    assert(hasInverse());
    float detInv = 1 / det();
    Vector3D lInvRow1(
        (row(1).y() * row(2).w() - row(1).w() * row(2).y()) * detInv,
        (row(0).w() * row(2).y() - row(0).y() * row(2).w()) * detInv,
        (row(0).y() * row(1).w() - row(0).w() * row(1).y()) * detInv);
    Vector3D lInvRow2(
        (row(1).w() * row(2).x() - row(1).x() * row(2).w()) * detInv,
        (row(0).x() * row(2).w() - row(0).w() * row(2).x()) * detInv,
        (row(0).w() * row(1).x() - row(0).x() * row(1).w()) * detInv);
    Vector3D lInvRow3(
        (row(1).x() * row(2).y() - row(1).y() * row(2).x()) * detInv,
        (row(0).y() * row(2).x() - row(0).x() * row(2).y()) * detInv,
        (row(0).x() * row(1).y() - row(0).y() * row(1).x()) * detInv);
    return Matrix3x3(lInvRow1, lInvRow2, lInvRow3);
}

// Output Operator
std::ostream& operator<<(std::ostream& os, const Matrix3x3& matrix) {
    os << "Matrix 3x3:" << endl;
    os << matrix.row(0).toString() << endl
        << matrix.row(1).toString() << endl
        << matrix.row(2).toString() << endl;
    return os;
}
```