

Swinburne University of Technology*School of Science, Computing and Engineering Technologies***ASSIGNMENT COVER SHEET**

Subject Code: COS30008
Subject Title: Data Structures and Patterns
Assignment number and title: 1, Solution Design in C++
Due date: Sunday, March 30, 2025, 23:59
Lecturer: Dr. Markus Lumpe

Your name: _____ **Your student ID:** _____

Marker's comments:

Problem	Marks	Obtained
1	38	
2	170	
Total	208	

Extension certification:

This assignment has been given an extension and is now due on _____

Signature of Convener: _____

```

//
//  Vector3D_PS1.cpp
//  problemset1
//
//  Created by Xinzhe Yu on 23/3/2025.
//
#include <iomanip>
#include "Vector3D.h"
#include <sstream>

bool Vector3D::operator==(const Vector3D &aOther) const noexcept{
    // check the difference of x, y and w between 'this' and aOther vector
    return this->x() - aOther.x() < std::numeric_limits<float>::epsilon()
        && std::fabs(this->y() - aOther.y()) <
        std::numeric_limits<float>::epsilon() && std::fabs(this->w() -
        aOther.w()) < std::numeric_limits<float>::epsilon();
}

std::string Vector3D::toString() const noexcept{
    // define a stringstream called result as the container to hold the
    string
    std::stringstream result;
    result << "[" << this->x() << "," << this->y() << "," << this->
    w() << "];"
    return result.str();
}

```

```

//
// Matrix3x3_PS1.cpp
// problemset1
//
// Created by Xinzhe Yu on 23/3/2025.
//

#include "Matrix3x3.h"
#include <cassert>
#include <sstream>

bool Matrix3x3::operator==(const Matrix3x3 &aOther) const noexcept{
    const Matrix3x3& M = *this;
    return (M[0] == aOther.fRows[0]) && (M[1] == aOther.fRows[1]) && (M[2]
        == aOther.fRows[2]);
}

Matrix3x3 Matrix3x3::operator*(const Matrix3x3& aOther) const noexcept {
    // the multiplication of 2 3x3 matrix can be calculated by using M[x]
    // to get each horizontal vector of the first matrix and using column(x)
    // to get each vertical vector of the second matrix, then using dot() to
    // get the new x y w for the new vector inside of the result matrix
    const Matrix3x3& M = *this;
    return Matrix3x3(
        Vector3D(M[0].dot(aOther.column(0)), M[0].dot(aOther.column(1)),
            M[0].dot(aOther.column(2))),
        Vector3D(M[1].dot(aOther.column(0)), M[1].dot(aOther.column(1)),
            M[1].dot(aOther.column(2))),
        Vector3D(M[2].dot(aOther.column(0)), M[2].dot(aOther.column(1)),
            M[2].dot(aOther.column(2)))
    );
}

Matrix3x3 Matrix3x3::transpose() const noexcept{
    const Matrix3x3& M = *this;
    return Matrix3x3(M.column(0), M.column(1), M.column(2));
}

float Matrix3x3::det() const noexcept {
    const Matrix3x3& M = *this;
    return M[0].x() * (M[1].y() * M[2].w() - M[1].w() * M[2].y()) -
        M[0].y() * (M[1].x() * M[2].w() - M[1].w() * M[2].x()) +
        M[0].w() * (M[1].x() * M[2].y() - M[1].y() * M[2].x());
}

bool Matrix3x3::hasInverse() const noexcept{
    // check whether the determination of this matrix is more than the
    // smallest difference with 0
    return std::fabs(det()) > std::numeric_limits<float>::epsilon();
}

Matrix3x3 Matrix3x3::inverse() const noexcept {
    assert(hasInverse());
    const Matrix3x3& M = *this;

```

```

// the inverse of a 3x3 matrix is equal to the inverse of its det
// multiple with Adjugate Matrix
// the inverse of its det can be calculated directly by det()
float invDet = 1.0f / det();
return Matrix3x3(
    Vector3D((M[1][1]*M[2][2]-M[1][2]*M[2][1])*invDet,
        (M[0][2]*M[2][1]-M[0][1]*M[2][2])*invDet,
        (M[0][1]*M[1][2]-M[0][2]*M[1][1])*invDet),
    Vector3D((M[1][2]*M[2][0]-M[1][0]*M[2][2])*invDet,
        (M[0][0]*M[2][2]-M[0][2]*M[2][0])*invDet,
        (M[0][2]*M[1][0]-M[0][0]*M[1][2])*invDet),
    Vector3D((M[1][0]*M[2][1]-M[1][1]*M[2][0])*invDet,
        (M[0][1]*M[2][0]-M[0][0]*M[2][1])*invDet,
        (M[0][0]*M[1][1]-M[0][1]*M[1][0])*invDet)
);
}

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

```

Result:

The screenshot shows a C++ IDE with a project named 'problemset1'. The code in the main file defines a `Vector3D` struct and a `Matrix3x3` struct. The `Vector3D` struct has a `toString` method and a `Vector3D` constructor. The `Matrix3x3` struct has a `toString` method and a `Matrix3x3` constructor. The output of the program is shown in the console, displaying the results of various operations on `Vector3D` and `Matrix3x3` objects.

```

3 // problemset1
4 //
5 // Created by Xinzhe Yu on 23/3/2025.
6 //
7 #include <iomanip>
8 #include "Vector3D.h"
9 #include <sstream>
10
11 bool Vector3D::operator==(const Vector3D &aOther) const noexcept{
12     // check the difference of x, y and w between 'this' and aother vector
13     return this->x() < aOther.x() < std::numeric_limits<float>::epsilon() && std::fabs(this->y() - aOther.y()) <
14         std::numeric_limits<float>::epsilon() && std::fabs(this->w() - aOther.w()) < std::numeric_limits<float>::epsilon();
15 }
16
17 int main() {
18     a == aa: true
19     a == b: false
20     a == c: false
21     b == c: false
22     a == a: true
23     b == b: true
24     c == c: true
25     Vector aa: [1,2,3]
26     Vector a: [1,2,3]
27     Vector b: [3.14159,3.14159,3.14159]
28     Vector c: [1.23457,9.87654,12435.1]
29     Matrix M1 is a rotation matrix.
30     det M = 1
31     Test matrix M2:
32     [[25,-3,-8],[6,2,15],[11,-3,4]]
33     M2 * M2 =
34     [[519,-57,-277],[327,-59,42],[381,-51,-117]]
35     det M2 = 1222
36     Does M2 have an inverse? Yes
37     transpose of M2:
38     [[25,6,11],[-3,2,-3],[-8,15,4]]
39     inverse of M2:
40     [[0.0433715,0.0294599,-0.0237316],[0.115385,0.153846,-0.346154],
41     [-0.0327332,0.0343699,0.0556465]]
42     inverse of M2 * 45:
43     [[1.95172,1.3257,-1.06792],[5.19231,6.92388,-15.5769],[-1.473,1.54664,2.58489]]
44     2 Test(s) completed.
45     Program ended with exit code: 0

```