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: Dhanveer Ramnauth **Your student ID:** 103866373

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:____________________

Figure 1: Vector3D_PS1.cpp

```cpp
#include <sstream>
#include "Vector3D.h"
#include <cmath>

bool Vector3D::operator==( const Vector3D& aOther ) const
 ↪  noexcept
{
    return std::abs(x() - aOther.x()) < eps &&
           std::abs(y() - aOther.y()) < eps &&
           std::abs(w() - aOther.w()) < eps;
}

std::string Vector3D::toString() const noexcept
{
    std::stringstream ss;
    ss << "[" << x() << "," << y() << "," << w() << "]";
    return ss.str();
}

```

Figure 2: Matrix3x3_PS1.cpp

```cpp
#include "Matrix3x3.h"
#include <cassert>

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

Matrix3x3 Matrix3x3::operator*( const Matrix3x3& aOther ) const
 ↪ noexcept
{
    const Matrix3x3& M = *this;
    const Vector3D& col1 = aOther.column(0);
    const Vector3D& col2 = aOther.column(1);
    const Vector3D& col3 = aOther.column(2);
    return Matrix3x3
    (
        { M[0].dot(col1), M[0].dot(col2), M[0].dot(col3) },
        { M[1].dot(col1), M[1].dot(col2), M[1].dot(col3) },
        { M[2].dot(col1), M[2].dot(col2), M[2].dot(col3) }
    );
}

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][0] * (M[1][1] * M[2][2] - M[1][2] * M[2][1]) -
        M[0][1] * (M[1][0] * M[2][2] - M[1][2] * M[2][0]) +
        M[0][2] * (M[1][0] * M[2][1] - M[1][1] * M[2][0]);
}
```

```

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

Matrix3x3 Matrix3x3::inverse() const noexcept
{

    assert(hasInverse());

    const Matrix3x3& M = *this;
    float reciprocal = 1 / det();

    return Matrix3x3
    (
    //                                          /   /                                          /
        { M[1][1] * M[2][2] - M[1][2] * M[2][1], M[0][2] * M[2][1] - M[0][1] * M[2][2], M[0][1] * M[1][2] - M[0][2] * M[1][1]},
        { M[1][2] * M[2][0] - M[1][0] * M[2][2], M[0][0] * M[2][2] - M[0][2] * M[2][0], M[0][2] * M[1][0] - M[0][0] * M[1][2]},
        { M[1][0] * M[2][1] - M[1][1] * M[2][0], M[0][1] * M[2][0] - M[0][0] * M[2][1], M[0][0] * M[1][1] - M[0][1] * M[1][0]}
    ) * reciprocal;

}

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