```
...oang\source\repos\Problem1\Problem1\Matrix3x3_PS1.cpp
```

```
1
```

```
1 #include "Matrix3x3_PS1.h"
 2
 3 Matrix3x3 Matrix3x3::operator*(const Matrix3x3& a0ther) const noexcept {
 4
       return Matrix3x3(
            Vector3D(row(0).dot(a0ther.column(0)), row(0).dot(a0ther.column
              (1)), row(0).dot(a0ther.column(2))),
            Vector3D(row(1).dot(a0ther.column(0)), row(1).dot(a0ther.column
 6
              (1)), row(1).dot(a0ther.column(2))),
 7
            Vector3D(row(2).dot(a0ther.column(0)), row(2).dot(a0ther.column
                                                                                 P
              (1)), row(2).dot(a0ther.column(2)))
       );
 8
 9 }
10
11 std::ostream& operator<<(std::ostream& aOStream, const Matrix3x3& aMatrix) >
       size_t lIndex = 0;
12
13
       aOStream << "[";
       while (lIndex < 3) {</pre>
14
15
            aOStream << aMatrix.row(lIndex).toString();
            if(lIndex++ !=2) aOStream << ",";</pre>
16
17
       }
       aOStream << "]";
18
19
       return aOStream;
20 }
21
22
23 float Matrix3x3::det() const noexcept {
24
       const Vector3D& row0 = row(0);
25
       const Vector3D& row1 = row(1);
26
       const Vector3D& row2 = row(2);
27
       return row0.x() * (row1.y() * row2.w() - row1.w() * row2.y())
28
29
             - row0.y() * (row1.x() * row2.w() - row1.w() * row2.x())
30
             + row0.w() * (row1.x() * row2.y() - row1.y() * row2.x());
31 }
32
33 bool Matrix3x3::hasInverse() const noexcept {
34
       return det() != 0.0f;
35 }
36
37 Matrix3x3 Matrix3x3::transpose() const noexcept {
38
       return Matrix3x3(
39
            Vector3D(column(0)[0], column(0)[1], column(0)[2]),
            Vector3D(column(1)[0], column(1)[1], column(1)[2]),
40
           Vector3D(column(2)[0], column(2)[1], column(2)[2])
41
42
       );
43 }
44
45 Matrix3x3 Matrix3x3::inverse() const noexcept {
```

```
...oang\source\repos\Problem1\Problem1\Matrix3x3_PS1.cpp
```

```
2
```

```
46
        if (det() != 0.0f) {
47
48
           float fInverseDetM = 1.0f / det();
49
50
           const Vector3D& row0 = row(0);
51
            const Vector3D& row1 = row(1);
            const Vector3D& row2 = row(2);
52
53
           float fInverseElement00 = (row1.y() * row2.w() - row1.w() * row2.y >
54
              ()) * fInverseDetM;
           float fInverseElement01 = -(row0.y() * row2.w() - row0.w() * row2.y →
55
              ()) * fInverseDetM;
            float fInverseElement02 = (row0.y() * row1.w() - row0.w() * row1.y >
56
              ()) * fInverseDetM;
57
           float fInverseElement10 = -(row1.x() * row2.w() - row1.w() * row2.x →
58
              ()) * fInverseDetM;
            float fInverseElement11 = (row0.x() * row2.w() - row0.w() * row2.x →
59
              ()) * fInverseDetM;
           float fInverseElement12 = -(row0.x() * row1.w() - row0.w() * row1.x →
60
              ()) * fInverseDetM;
61
62
            float fInverseElement20 = (row1.x() * row2.y() - row1.y() * row2.x >
              ()) * fInverseDetM;
           float fInverseElement21 = -(row0.x() * row2.y() - row0.y() * row2.x →
63
              ()) * fInverseDetM;
            float fInverseElement22 = (row0.x() * row1.y() - row0.y() * row1.x >
64
              ()) * fInverseDetM;
65
           return Matrix3x3(Vector3D(fInverseElement00, fInverseElement01,
66
              fInverseElement02),
                             Vector3D(fInverseElement10, fInverseElement11,
67
                       fInverseElement12),
68
                             Vector3D(fInverseElement20, fInverseElement21,
                       fInverseElement22));
       }
69
70
       else {
71
            std::cout << "Determinate of M is Zero";</pre>
       }
72
73 }
74
75
76
77
78
79
80
81
82
```