Yifu He_ Homework 4_ Test Case
190003956

**Constructors:**

```
M1:
0   0   0
0   0   0
0   0   0

M2(2):
0   0
0   0

M3(2,3):
0   0   0
0   0   0
```

```
M4(3,3,1.0):
1   1   1
1   1   1
1   1   1
```

```
M5(vec_B):
0   0   1
0   1   0
1   0   0
```

**Transpose:**

```
M3(2,3):
0   0   0
0   0   0

M3.Transpose()
0   0
0   0
0   0
```

**Operator + - ***

```
M4(3,3,1.0):
1   1   1
1   1   1
1   1   1
```

```
M5(vec_B):
0   0   1
0   1   0
1   0   0
```

+:

```
M4 + M5:
1   1   2
1   2   1
2   1   1
```

-:

```
M4 - M5:
1   1   0
1   0   1
0   1   1
```

*:

```
M4 * 2:
2   2   2
2   2   2
2   2   2

2 * M4:
2   2   2
2   2   2
2   2   2
```

```
M4 * M5:
1   1   1
1   1   1
1   1   1

M4 * M4:
3   3   3
3   3   3
3   3   3
```

**Assign constructor:**

```
Matrix M6(M5);
0   0   1
0   1   0
1   0   0
```

**Assign operator:**

```
M6 = M4;
1   1   1
1   1   1
1   1   1
```

**Destructor:**

Cause I didn't use pointer in this assignment, so this is no test case. I just put me code here

```
// Rule of Three: copy constructor, assign constructor, destructor
Matrix(const Matrix& mtx);
Matrix& operator=(const Matrix& mtx);
~Matrix(){}
```

**Operator >> and <<:**

```
Show operator >>:
the Matrix is a 3 * 3 Matrix.
Input your matrix below:
1 2 3
3 2 1
1.5 2.5 3.5

Show operator <<, print the Matrix we just input
1   2   3
3   2   1
1.5 2.5 3.5
```

**Function identity matrix:**

```
M7
1   12  10  -12
0   1   0   12
0   0   -1  4
0   0   1   -3
```

```
M7.Identity():
1   0   0   0
0   1   0   0
0   0   1   0
0   0   0   1
```

**Extra credit**

**Function inverse:**

```
M7
1   12  10  -12
0   1   0   12
0   0   -1  4
0   0   1   -3
```

```
M7.Inverse():
1   -12 126 116
0   1   -12 -12
-0  0   3   4
0   -0  1   1
```

**Test M7^(-1) * M7:**

```
M7^(-1) * M7:
1   0   0   0
0   1   0   0
0   0   1   0
0   0   0   1
```

**Extra function:**

**Function Deteminant:**

**Please use getA() !!!**

```
M4(3,3,1.0):
1   1   1
1   1   1
1   1   1
```

```
M5(vec_B):
0   0   1
0   1   0
1   0   0
```

```
M4.getA()
0

M5.Determinant()
-1
```

```
M7
1   12  10  -12
0   1   0   12
0   0   -1  4
0   0   1   -3
M7.Determinant(): -1  ---- (calculate |A| in elementary transformation way.)
M7.getA(): -1  ---- (calculate |A| by sum(a[0][i] * A(0,i)) i from 0 to n-1)
```

**Function Adjoint Matrix:**

```
M7.getAdjointMatrix():
-1  12  -126    -116
-0  -1  12  12
0   -0  -3  -4
-0  0   -1  -1
```

**Test it with `(1/|M7|) * M7(adjoint) * M7:`**

```
(1/|M7|) * M7(adjoint) * M7:
1   0   0   0
0   1   0   0
0   0   1   0
0   0   0   1
```

**Abnormal value error handling:**

1. 
```
M2(2):
0   0
0   0
```

```
M5(vec_B):
0   0   1
0   1   0
1   0   0
```

```
M2 * M5:
Error: Cannot perform multiplication in those two Matrix!
```

2.
```
M2 + M5:
Error: Two Matrix have different dimention!
```

3.
```
M3(2,3):
0   0   0
0   0   0
```

```
M3.Inverse()
Error: Matix is not square! It has no inversed Matrix
```

4.
```
M4(3,3,1.0):
1   1   1
1   1   1
1   1   1
```

```
M4.Inverse()
Error: Matrix is singular! It has no inversed Matrix
```