myMatrix.h

```
#include "../../include/myMatrix.h"
```

자신의 폴더에 위치에 있는 myMatrix.h라는 헤더파일 가져오는 방법

```
#include <iostream>
#include <string>
#include <fstream>
```

```
typedef struct {
    double** at;//2d array, 더블 포인터
    int rows, cols;//dimension 정보 저장
}Matrix;
```

행렬을 표현할 구조체 선언해주기, 앞으로 이런 형식으로 행렬을 선언해 줄 것이다.

행렬을 가져올 때 기본적으로 지켜야 될 규칙들

```
#define ASGN        999     // enter your assignment number
#define EVAL        0       // [¡Ø DO NOT EDIT !!!]

#include "../../include/myMatrix.h"
int main(int argc, char* argv[])
{
    /*   [¡Ø DO NOT EDIT !!!]   Resources file path setting for evaluation  */
    std::string path = "C:/NP_data/Assignment" + std::to_string(ASGN) + "/";


#if EVAL
    path += "eval/";
#endif
    return 0;
}
```

Solve linear equation

- Gauss_elimation()
- Gauss_Jordan_elimation()
- gaussElim()
- backsub()
- backsub_1()
- fwdsub()
- LUdecomp_2()
- solveLU_1()
- invMat()

- invMat_1()
- LUdecomp_1()
- LUdecomp()
- solveLU()

Gauss_elimation()

Gauss elimination을 실시한다. 이때 첨가행렬를 만들고 Gauss elimination을 한다.

해를 구할 때 back substitution의 방식을 이용해서 구한다.

Gauss elimination을 실시한 행렬과 해를 출력한다.

```
Matrix  Gauss_elimation(Matrix _A,Matrix _B);
```

Parameters

> _A: 첨가행렬을 만들때 왼쪽에 위치해있는 행렬
>
> _B: 첨가행렬을 만들때 오른쪽에 위치해있는 행렬

Example code

```
#define ASGN        999     // enter your assignment number
#define EVAL        0       // [¡Ø DO NOT EDIT !!!]

#include "../../include/myMatrix.h"
int main(int argc, char* argv[])
{
    /*   [¡Ø DO NOT EDIT !!!]   Resources file path setting for evaluation  */
    std::string path = "C:/NP_data/Assignment" + std::to_string(ASGN) + "/";


#if EVAL
    path += "eval/";
#endif
    Matrix matA = txt2Mat(path, "prob1_matA");
    Matrix vecb = txt2Mat(path, "prob1_vecb");
    Matrix matTemp = Gauss_elimation(matA, vecb);

    printMat(matA, "matA");
    printMat(vecb, "vecb");
    printMat(matTemp, "Gauss elimation of matA");

    return 0;
}
```

output

```
matA =
       4.000000       -2.000000       -3.000000        6.000000
      -6.000000        7.000000        6.500000       -6.000000
       1.000000        7.500000        6.250000        5.500000
     -12.000000       22.000000       15.500000       -1.000000

vecb =
      12.000000
```

```
      -6.500000
      16.000000
      17.000000

Gauss elimation of  matA =
       4.000000       -2.000000       -3.000000        6.000000       12.000000
       0.000000        4.000000        2.000000        3.000000       11.500000
       0.000000        0.000000        3.000000       -2.000000      -10.000000
       0.000000        0.000000        0.000000        4.000000        2.000000

Solution of x =
       2.000000
       4.000000
      -3.000000
       0.500000
```

Warning

- _A, _B는 행렬이어야 한다.
- _B는 n X 1 행렬이어야 한다.

Error Handling

- 두 행렬의 행의 전체 개수가 같지 않으면 오류가 발생한다.

Gauss_Jordan_elimation()

Gauss Jordan elimination을 실시한다. 이때 첨가행렬를 만들고 Gauss Jordan elimination을 한다.

Gauss Jordan elimination은 pivot은 1, 나머지 요소들은 0으로 만드는 것을 의미한다.

pivot을 제외한 나머지 요소들은 0이기 때문에 해를 구할 때 제일 오른쪽 항을 보면 된다.

Gauss Jordan elimination을 실시한 행렬과 해를 출력한다.

```
Matrix  Gauss_Jordan_elimation(Matrix _A, Matrix _B);
```

Parameters

> _A: 첨가행렬을 만들때 왼쪽에 위치해있는 행렬
>
> _B: 첨가행렬을 만들때 오른쪽에 위치해있는 행렬

Example code

```
#define ASGN        999     // enter your assignment number
#define EVAL        0       // [¡Ø DO NOT EDIT !!!]

#include "../../include/myMatrix.h"
int main(int argc, char* argv[])
{
    /*   [¡Ø DO NOT EDIT !!!]   Resources file path setting for evaluation  */
    std::string path = "C:/NP_data/Assignment" + std::to_string(ASGN) + "/";


#if EVAL
    path += "eval/";
```

```
#endif
    Matrix matA = txt2Mat(path, "prob1_matA");
    Matrix vecb = txt2Mat(path, "prob1_vecb");
    Matrix matTemp = Gauss_Jordan_elimation(matA, vecb);

    printMat(matA, "matA");
    printMat(vecb, "vecb");
    printMat(matTemp, "Gauss Jordan elimation of matA");

    return 0;
}
```

output

```
matA =
       4.000000       -2.000000       -3.000000        6.000000
      -6.000000        7.000000        6.500000       -6.000000
       1.000000        7.500000        6.250000        5.500000
     -12.000000       22.000000       15.500000       -1.000000

vecb =
      12.000000
      -6.500000
      16.000000
      17.000000

Gauss Jordan elimation of matA =
       1.000000        0.000000        0.000000        0.000000        2.000000
       0.000000        1.000000        0.000000        0.000000        4.000000
       0.000000        0.000000        1.000000        0.000000       -3.000000
       0.000000        0.000000        0.000000        1.000000        0.500000
```

Warning

- _A, _B는 행렬이어야 한다.
- _B는 n X 1 행렬이어야 한다.

Error Handling

- 두 행렬의 행의 전체 개수가 같지 않으면 오류가 발생한다.

gaussElim()

Gauss elimination을 실시한다.

Gauss elimination을 실시한 행렬과 해를 출력한다.

이때 pivoting을 이용해서 Gauss elimination을 실시한다. 이때 scale partial pivoting을 실시한다.

|pivot|/|해당열에서 절대값이 제일 큰 값|이 제일 큰 행과 위치를 바꾼다.

```
void gaussElim(Matrix A, Matrix b, Matrix U, Matrix d, Matrix P);
```

Parameters

> A: Gauss eliminatin을 실시할 때 왼쪽에 위치해 있는 행렬 (n x n)

> B: Gauss eliminatin을 실시할 때 오른쪽에 위치해 있는 행렬 (n x 1)
>
> U: Gauss eliminatin을 실시한 A행렬의 결과 (n x n)
>
> d: Gauss eliminatin을 실시한 b행렬의 결과 (n x 1)
>
> P: pivoting을 하기 위해 필요한 행렬 (n x n)

Example code

```
#define ASGN        999     // enter your assignment number
#define EVAL        0       // [¡Ø DO NOT EDIT !!!]

#include "../../include/myMatrix.h"
int main(int argc, char* argv[])
{
    /*   [¡Ø DO NOT EDIT !!!]   Resources file path setting for evaluation  */
    std::string path = "C:/NP_data/Assignment" + std::to_string(ASGN) + "/";


#if EVAL
    path += "eval/";
#endif
    Matrix matA = txt2Mat(path, "prob1_matA");
    Matrix vecb = txt2Mat(path, "prob1_vecb");
    Matrix matU= createMat(matA.rows, matA.cols);
    Matrix matd = createMat(vecb.rows, vecb.cols);
    gaussElim(matA, vecb, matU, matd, matP);

    printMat(matA, "matA");
    printMat(vecb, "vecb");
    printMat(matU, "matU");
    printMat(matd, "matd");

    return 0;
}
```

output

```
matA =
       9.500000       -2.500000        0.000000       -2.000000        0.000000
      -2.500000       11.000000       -3.500000        0.000000       -5.000000
       0.000000       -3.500000       15.500000        0.000000       -4.000000
      -2.000000        0.000000        0.000000        7.000000       -3.000000
       0.000000       -5.000000       -4.000000       -3.000000       12.000000

vecb =
      12.000000
     -16.000000
      14.000000
      10.000000
     -30.000000

matU =
       9.500000       -2.500000        0.000000       -2.000000        0.000000
       0.000000       10.342105       -3.500000       -0.526316       -5.000000
       0.000000        0.000000       14.315522       -0.178117       -5.692112
       0.000000        0.000000        0.000000        6.549947       -3.325276
```

```
       0.000000        0.000000        0.000000        0.000000        5.631235

matd =
      12.000000
     -12.842105
       9.653944
      11.992890
     -26.281520
```

Warning

- A는 n x n 행렬, B는 n x1 행렬이어야 한다.

Error Handling

- A가 정방행렬이 아니면 오류가 발생한다.
- gauss elimination을 할 때 0으로 나누면 오류가 발생한다.
- b행렬과 d행렬의 열이 1이 아니면 오류가 발생한다
- A행렬의 크기가 U행렬 혹은 P행렬의 크기와 같지 않으면 오류가 발생한다.
- b행렬과 d행렬의 크기가 같지 않으면 오류가 발생한다.
- b, d행렬의 행의 개수는 A행렬의 행의 개수와 같아야 한다.

backsub()

U가 위삼각 행렬이기 때문에 back substitution을 이용해 해를 구한다.

$$x_i = \frac{1}{a_{ii}}(y_i - \sum_{j=i+1}^{n} a_{ij}x_j)$$

```
void backsub(Matrix U, Matrix y, Matrix x);
```

Parameters

> U: 위삼각 행렬이어야 한다. 방정식의 계수를 모아 놓은 행렬
>
> y: 방정식의 결과 혹은 입력값들을 모아 놓은 행렬 (n x1)
>
> x: 원하는 해를 모아놓은 행렬(n x 1)

Example code

```
#define ASGN        999     // enter your assignment number
#define EVAL        0       // [¡Ø DO NOT EDIT !!!]

#include "../../include/myMatrix.h"
int main(int argc, char* argv[])
{
    /*   [¡Ø DO NOT EDIT !!!]   Resources file path setting for evaluation  */
    std::string path = "C:/NP_data/Assignment" + std::to_string(ASGN) + "/";


#if EVAL
    path += "eval/";
#endif
    Matrix matA = txt2Mat(path, "prob1_matA");
    Matrix vecb = txt2Mat(path, "prob1_vecb");
```

```
    Matrix matU= createMat(matA.rows, matA.cols);
    Matrix matd = createMat(vecb.rows, vecb.cols);
    Matrix matX = createMat(vecb.rows, vecb.cols);

    gaussElim(matA, vecb, matU, matd, matP);
    backsub(matU, matd, matX);

    printMat(matA, "matA");
    printMat(vecb, "vecb");
    printMat(matU, "matU");
    printMat(matd, "matd");
    printMat(matX, "matX");
    return 0;
}
```

output

```
matA =
       9.500000       -2.500000        0.000000       -2.000000        0.000000
      -2.500000       11.000000       -3.500000        0.000000       -5.000000
       0.000000       -3.500000       15.500000        0.000000       -4.000000
      -2.000000        0.000000        0.000000        7.000000       -3.000000
       0.000000       -5.000000       -4.000000       -3.000000       12.000000

vecb =
      12.000000
     -16.000000
      14.000000
      10.000000
     -30.000000

matU =
       9.500000       -2.500000        0.000000       -2.000000        0.000000
       0.000000       10.342105       -3.500000       -0.526316       -5.000000
       0.000000        0.000000       14.315522       -0.178117       -5.692112
       0.000000        0.000000        0.000000        6.549947       -3.325276
       0.000000        0.000000        0.000000        0.000000        5.631235

matd =
      12.000000
     -12.842105
       9.653944
      11.992890
     -26.281520

matX =
       0.116244
      -3.927551
      -1.188053
      -0.538400
      -4.667097
```

Warning

- U는 위삼각행렬, y, x는 n x 1 행렬이어야 한다.

Error Handling

- U가 정방행렬이어야 한다.
- y, x 행렬의 크기는 같아야 한다.
- y, x 행렬의 행의 개수는 U 행렬의 행의 개수와 같아야 한다.
- y, x 행렬의 열의 개수는 1이다.

backsub_1()

U가 위삼각 행렬이기 때문에 back substitution을 이용해 해를 구한다.

기존의 backsub()함수도 역할은 똑같다. 파라미터가 y에서 d로 이름이 바뀌었다.

$$x_i = \frac{1}{a_{ii}}(d_i - \sum_{j=i+1}^{n} a_{ij}x_j)$$

```
void backsub_1(Matrix U, Matrix d, Matrix x);
```

Parameters

> U: 위삼각 행렬이어야 한다. 방정식의 계수를 모아 놓은 행렬
>
> d: 방정식의 결과 혹은 입력값들을 모아 놓은 행렬 (n x1)
>
> x: 원하는 해를 모아놓은 행렬(n x 1)

Example code

```
#define ASGN        999     // enter your assignment number
#define EVAL        0       // [¡Ø DO NOT EDIT !!!]

#include "../../include/myMatrix.h"
int main(int argc, char* argv[])
{
    /*   [¡Ø DO NOT EDIT !!!]   Resources file path setting for evaluation  */
    std::string path = "C:/NP_data/Assignment" + std::to_string(ASGN) + "/";


#if EVAL
    path += "eval/";
#endif
    Matrix matA = txt2Mat(path, "prob1_matA");
    Matrix vecb = txt2Mat(path, "prob1_vecb");
    Matrix matU= createMat(matA.rows, matA.cols);
    Matrix matd = createMat(vecb.rows, vecb.cols);
    Matrix matX = createMat(vecb.rows, vecb.cols);

    gaussElim(matA, vecb, matU, matd, matP);
    backsub_1(matU, matd, matX);

    printMat(matA, "matA");
    printMat(vecb, "vecb");
    printMat(matU, "matU");
    printMat(matd, "matd");
    printMat(matX, "matX");
    return 0;
}
```

output

```
matA =
       9.500000       -2.500000        0.000000       -2.000000        0.000000
      -2.500000       11.000000       -3.500000        0.000000       -5.000000
       0.000000       -3.500000       15.500000        0.000000       -4.000000
      -2.000000        0.000000        0.000000        7.000000       -3.000000
       0.000000       -5.000000       -4.000000       -3.000000       12.000000

vecb =
      12.000000
     -16.000000
      14.000000
      10.000000
     -30.000000

matU =
       9.500000       -2.500000        0.000000       -2.000000        0.000000
       0.000000       10.342105       -3.500000       -0.526316       -5.000000
       0.000000        0.000000       14.315522       -0.178117       -5.692112
       0.000000        0.000000        0.000000        6.549947       -3.325276
       0.000000        0.000000        0.000000        0.000000        5.631235

matd =
      12.000000
     -12.842105
       9.653944
      11.992890
     -26.281520

matX =
       0.116244
      -3.927551
      -1.188053
      -0.538400
      -4.667097
```

Warning

- U는 위삼각행렬, d, x는 n x 1 행렬이어야 한다.

Error Handling

- U가 정방행렬이어야 한다.
- d, x 행렬의 크기는 같아야 한다.
- d, x 행렬의 행의 개수는 U 행렬의 행의 개수와 같아야 한다.
- d, x 행렬의 열의 개수는 1이다.

fwdsub()

U가 아래삼각 행렬이기 때문에 forward substitution을 이용해 해를 구한다.

$$x_i = \frac{1}{a_{ii}}(b_i - \sum_{j=1}^{i-1} a_{ij}y_j)$$

```
void fwdsub(Matrix L, Matrix b, Matrix y);
```

Parameters

> L: 아래삼각 행렬이어야 한다. 방정식의 계수를 모아 놓은 행렬
>
> b: 방정식의 결과 혹은 입력값들을 모아 놓은 행렬 (n x1)
>
> y: 원하는 해를 모아놓은 행렬(n x 1)

Example code

```
#define ASGN        999     // enter your assignment number
#define EVAL        0       // [¡Ø DO NOT EDIT !!!]

#include "../../include/myMatrix.h"
int main(int argc, char* argv[])
{
    /*   [¡Ø DO NOT EDIT !!!]   Resources file path setting for evaluation  */
    std::string path = "C:/NP_data/Assignment" + std::to_string(ASGN) + "/";


#if EVAL
    path += "eval/";
#endif
    Matrix matK = txt2Mat(path, "prob1_matK");
    Matrix vecd = txt2Mat(path, "prob1_vecf");
    Matrix matU = zeros(matK.rows, matK.cols);
    Matrix matL = eye(matK.rows, matK.cols);
    Matrix  matP= eye(matK.rows, matK.cols);

    LUdecomp(matK, matL, matU,matP);
    Matrix matY = createMat(vecd.rows, vecd.cols);

    fwdsub(matL, vecd, matY);
    printMat(matL, "matL");
    printMat(matU, "matU");
    printMat(matY, "matY");

    return 0;
}
```

output

```
matL =
       1.000000        0.000000        0.000000
      -0.266667        1.000000        0.000000
       0.000000       -0.505618        1.000000

matU =
      75.000000      -20.000000        0.000000
       0.000000       29.666667      -15.000000
       0.000000        0.000000        7.415730

matY =
      19.620001
      34.662001
      32.240731
```

Warning

- L는 아래삼각 행렬, b, y는 n x1 행렬이어야 한다.

Error Handling

- L이 정방행렬이어야 한다.
- y, b 행렬의 크기는 같아야 한다.
- y, b 행렬의 행의 개수는 L 행렬의 행의 개수와 같아야 한다.
- y, b 행렬의 열의 개수는 1이다.

LUdecomp_2()

행렬 A를 L행렬과 U행렬로 분해를 한다. 이 함수는 pivoting을 고려하지 않고 LU분해를 실시한다.

$$Ax = B$$
$$A = LU$$

gauss elimination과 과정은 유사하다.

1. L은 초기에 단위행렬로 초기화해준다.
2. L은 gauss elimination을 할 때 pivot이 속해있는 열 중에서 pivot의 행보다 큰 행에 위치해있는 요소들을 pivot으로 나눈다. 그 값을 L행렬에 해당 위치에 옮긴다.
3. U는 gauss elimination에 의해 나온 행렬이다.

```
void LUdecomp_2(Matrix A, Matrix L, Matrix U);
```

Parameters

> A: n x n 행렬이다.
>
> L: 아래삼각 행렬이다. A를 LU분해했을 때 결과로 나온 행렬이다. (n x n)
>
> U: 위 삼각행렬이다. A를 LU분해했을 때 결과로 나온 행렬이다. (n x n)

Example code

```
#define ASGN        999     // enter your assignment number
#define EVAL        0       // [¡Ø DO NOT EDIT !!!]

#include "../../include/myMatrix.h"
int main(int argc, char* argv[])
{
    /*   [¡Ø DO NOT EDIT !!!]   Resources file path setting for evaluation  */
    std::string path = "C:/NP_data/Assignment" + std::to_string(ASGN) + "/";


#if EVAL
    path += "eval/";
#endif
    Matrix matK = txt2Mat(path, "prob1_matK");
    Matrix matU = zeros(matK.rows, matK.cols);
    Matrix matL = eye(matK.rows, matK.cols);
    LUdecomp_2(matK, matL, matU);

    printMat(matK, "matK");
    printMat(matL, "matL");
```

```
    printMat(matU, "matU");

    return 0;
}
```

output

```
matK =
      75.000000      -20.000000        0.000000
       0.000000       29.666667      -15.000000
       0.000000        0.000000        7.415730

matL =
       1.000000        0.000000        0.000000
      -0.266667        1.000000        0.000000
       0.000000       -0.505618        1.000000

matU =
      75.000000      -20.000000        0.000000
       0.000000       29.666667      -15.000000
       0.000000        0.000000        7.415730
```

Warning

- L는 아래삼각 행렬, U는 위삼각행렬이어야 한다.
- A: n x n 행렬이어야 한다.

Error Handling

- A가 정방행렬이 아니면 오류가 발생한다.
- A행렬의 크기가 U행렬 혹은 L행렬의 크기와 같지 않으면 오류가 발생한다.

solveLU_1()

pivoting을 고려하지 않고 LU분해한 것의 해를 구하는 것이다.

$$Ly = B$$
$$Ux = y$$

1. Ly=b 이 식을 푼다. 이때 L이 아래삼각행렬이므로 forward substitution을 이용해 해를 구한다.
2. Ux=y 이 식을 푼다. 이때 U가 위삼각행렬이므로 backward substitution을 이용해 해를 구한다.

```
void solveLU_1(Matrix L, Matrix U, Matrix b, Matrix x);
```

Parameters

> L: 아래삼각 행렬이다.
>
> U: 위 삼각행렬이다.
>
> b: 입력값 혹은 방정식의 결과인 행렬이다. n x 1 행렬이다.
>
> x: 최종적으로 원하는 해를 모은 행렬이다. n x 1 행렬이다.

Example code

```
#define ASGN        999     // enter your assignment number
#define EVAL        0       // [¡Ø DO NOT EDIT !!!]

#include "../../include/myMatrix.h"
int main(int argc, char* argv[])
{
    /*   [¡Ø DO NOT EDIT !!!]   Resources file path setting for evaluation  */
    std::string path = "C:/NP_data/Assignment" + std::to_string(ASGN) + "/";


#if EVAL
    path += "eval/";
#endif
    Matrix matK = txt2Mat(path, "prob1_matK");
    Matrix matU = zeros(matK.rows, matK.cols);
    Matrix matL = eye(matK.rows, matK.cols);
    Matrix matX = createMat(vecd.rows, vecd.cols);
    Matrix vecd = txt2Mat(path, "prob1_vecf");

    LUdecomp_2(matK, matL, matU);
    solveLU_1(matL, matU, vectd, matX);

    printMat(matK, "matK");
    printMat(matL, "matL");
    printMat(matU, "matU");
    printMat(matX, "matX");

    return 0;
}
```

output

```
matK =
      75.000000      -20.000000        0.000000
       0.000000       29.666667      -15.000000
       0.000000        0.000000        7.415730

matL =
       1.000000        0.000000        0.000000
      -0.266667        1.000000        0.000000
       0.000000       -0.505618        1.000000

matU =
      75.000000      -20.000000        0.000000
       0.000000       29.666667      -15.000000
       0.000000        0.000000        7.415730

matX =
       1.159364
       3.366614
       4.347614
```

Warning

- L는 아래삼각 행렬, U는 위삼각행렬이어야 한다.
- x, y: n x 1 행렬이어야 한다.

Error Handling

- L행렬과 U행렬의 사이즈는 같아야 한다.
- b행렬과 x행렬의 사이즈는 같아야 한다.
- b행렬과 x행렬의 행의 개수는 L, U행렬의 행의 개수와 같아야 한다.
- b행렬과 x행렬의 열의 개수는 1이어야 한다.

invMat()

역행렬을 구하는 행렬이다.

1. 행렬 A를 LU분해를 실시한다.
2. Ainv의 각 열들의 해를 계산한다.
3. 각 해를 해당 Ainv의 열로 옮긴다.

```
double invMat(Matrix A, Matrix Ainv);
```

Parameters

> A: 역행렬을 할 행렬
>
> Ainv: 역행렬 결과인 행렬

Example code

```
#define ASGN        999     // enter your assignment number
#define EVAL        0       // [¡Ø DO NOT EDIT !!!]

#include "../../include/myMatrix.h"
int main(int argc, char* argv[])
{
    /*   [¡Ø DO NOT EDIT !!!]   Resources file path setting for evaluation  */
    std::string path = "C:/NP_data/Assignment" + std::to_string(ASGN) + "/";


#if EVAL
    path += "eval/";
#endif
    Matrix matK = txt2Mat(path, "prob1_matK");
    Matrix invK = eye(matK.rows, matK.cols);
    invMat(matK, invK);
    printMat(invK, "invK");

    return 0;
}
```

output

```
matK =
      75.000000      -20.000000        0.000000
       0.000000       29.666667      -15.000000
       0.000000        0.000000        7.415730

invK =
       0.018182        0.018182        0.018182
       0.018182        0.068182        0.068182
       0.018182        0.068182        0.134848
```

Warning

- Ainv는 초기 단위행렬로 설정한다.
- A: n x n 행렬이어야 한다.
- Ainv와 A의 행렬의 크기는 같아야 한다.

Error Handling

- A와 Ainv의 행렬의 크기가 같지 않으면 오류가 발생한다.
- 정방행렬이 아니면 오류가 발생한다.
- 주대각선 요소성분에 0이 있으면 오류가 발생한다.

invMat_1()

역행렬을 구하는 행렬이다.

1. 행렬 A를 LU분해를 실시한다.
2. [1;0;0], [0;1;0], [0;0;1] 이런 식으로 함수 내에서 새롭게 계산할 행렬을 추가로 만들어서 역행렬을 구한다.
3. 각 해를 해당 Ainv의 열로 옮긴다.

```
double invMat_1(Matrix A, Matrix Ainv);
```

Parameters

> A: 역행렬을 할 행렬
>
> Ainv: 역행렬 결과인 행렬

Example code

```
#define ASGN        999     // enter your assignment number
#define EVAL        0       // [¡Ø DO NOT EDIT !!!]

#include "../../include/myMatrix.h"
int main(int argc, char* argv[])
{
    /*   [¡Ø DO NOT EDIT !!!]   Resources file path setting for evaluation  */
    std::string path = "C:/NP_data/Assignment" + std::to_string(ASGN) + "/";


#if EVAL
    path += "eval/";
#endif
    Matrix matK = txt2Mat(path, "prob1_matK");
```

```
    Matrix invK = eye(matK.rows, matK.cols);
    invMat_1(matK, invK);
    printMat(invK, "invK");

    return 0;
}
```

output

```
matK =
      75.000000      -20.000000        0.000000
       0.000000       29.666667      -15.000000
       0.000000        0.000000        7.415730

invK =
       0.018182        0.018182        0.018182
       0.018182        0.068182        0.068182
       0.018182        0.068182        0.134848
```

Warning

- Ainv는 초기 단위행렬로 설정한다.
- A: n x n 행렬이어야 한다.
- Ainv와 A의 행렬의 크기는 같아야 한다.

Error Handling

- A와 Ainv의 행렬의 크기가 같지 않으면 오류가 발생한다.
- 정방행렬이 아니면 오류가 발생한다.
- 주대각선 요소성분에 0이 있으면 오류가 발생한다.

LUdecomp()

A를 LU분해를 실시한다. 이때 scale partial pivoting을 이용한다.

L, P는 초기에 단위행렬로 초기화해준다.

1. A행렬에서 |pivot|/|해당열에서 절대값이 제일 큰 값|이 제일 큰 행과 위치를 바꾼다.
2. L은 gauss elimination을 할 때 pivot이 속해있는 열 중에서 pivot의 행보다 큰 행에 위치해있는 요소들을 pivot으로 나눈다. 그 값을 L행렬에 해당 위치에 옮긴다.
3. L도 A와 같이 행렬의 위치를 변경시킨다. 행렬을 변경시키는 A를 변경시키는 위치와 동일하다.
4. 이 반복을 계속 반복을 한다.
5. U는 gauss elimination에 의해 나온 행렬이다.

```
void LUdecomp(Matrix A, Matrix L, Matrix U, Matrix P);
```

Parameters

> A: LU 분해를 할 행렬
>
> L: pivoting을 고려해서 나온 아래 삼각행렬
>
> U: pivoting을 고려해서 나온 위삼각행렬
>
> P: pivoting을 고려하기 위한 행렬

Example code

```
#define ASGN        999     // enter your assignment number
#define EVAL        0       // [¡Ø DO NOT EDIT !!!]

#include "../../include/myMatrix.h"
int main(int argc, char* argv[])
{
    /*   [¡Ø DO NOT EDIT !!!]   Resources file path setting for evaluation  */
    std::string path = "C:/NP_data/Assignment" + std::to_string(ASGN) + "/";


#if EVAL
    path += "eval/";
#endif
    Matrix matK = txt2Mat(path, "prob1_matK");
    Matrix matU = zeros(matK.rows, matK.cols);
    Matrix matL = eye(matK.rows, matK.cols);
    Matrix matP= eye(matK.rows, matK.cols);
    LUdecomp(matK, matL, matU,matP);

    printMat(matK, "matK");
    printMat(matL, "matL");
    printMat(matU, "matU")
    return 0;
}
```

output

```
matK =
      75.000000      -20.000000        0.000000
     -20.000000       35.000000      -15.000000
       0.000000      -15.000000       15.000000

matL =
       1.000000        0.000000        0.000000
      -0.266667        1.000000        0.000000
       0.000000       -0.505618        1.000000

matU =
      75.000000      -20.000000        0.000000
       0.000000       29.666667      -15.000000
       0.000000        0.000000        7.415730
```

Warning

- L, P는 초기에 단위행렬이어야 한다.
- A: n x n 행렬이어야 한다.
- 계산결과 나온 L은 아래삼각행렬, U는 위삼각행렬이어야 한다.

Error Handling

- A, L, U, P 행렬이 정방행렬이 아니면 오류가 발생한다.
- 0으로 나눌 시 오류가 발생한다.

solveLU()

LU분해한 것의 해를 구한다. L과 U는 pivoting을 고려한 LU분해의 결과이다. Ax에 P를 곱했기 때문에 B에도 P를 곱한 후 해를 구한다. P는 pivoting을 고려하여 LU분해했을 때 결과로 나온 행렬이다. P는 LU분해를 하기 전에 단위행렬로 설정해주어야 한다.

$$PAx = PB$$
$$LUx = PB$$
$$Ly = PB$$
$$UX = y$$

1. Ly=PB 이 식을 푼다. 이때 L이 아래삼각행렬이므로 forward substitution을 이용해 해를 구한다.
2. Ux=y 이 식을 푼다. 이때 U가 위삼각행렬이므로 backward substitution을 이용해 해를 구한다.

```
void solveLU(Matrix L, Matrix U, Matrix b, Matrix x, Matrix P);
```

Parameters

> L: pivoting를 고려한 LU분해를 했을 때 나온 아래삼각행렬 (n x n)
>
> U: pivoting를 고려한 LU분해를 했을 때 나온 위삼각행렬 (n x n)
>
> b: 방정식들의 입력값 혹은 결과값들을 모아놓은 행렬 (n x 1)
>
> x: 방정식들의 해들을 모아놓은 행렬 (n x 1)
>
> P: 단위행렬로 초기화를 한 후 pivoting을 고려한 LU분해을 했을 때 나온 행렬이다. (n x n)

Example code

```
#define ASGN        999     // enter your assignment number
#define EVAL        0       // [¡Ø DO NOT EDIT !!!]

#include "../../include/myMatrix.h"
int main(int argc, char* argv[])
{
    /*   [¡Ø DO NOT EDIT !!!]   Resources file path setting for evaluation  */
    std::string path = "C:/NP_data/Assignment" + std::to_string(ASGN) + "/";


#if EVAL
    path += "eval/";
#endif
    Matrix matK = txt2Mat(path, "prob1_matK");
    Matrix vecd = txt2Mat(path, "prob1_vecf");
    Matrix matU = zeros(matK.rows, matK.cols);
    Matrix matL = eye(matK.rows, matK.cols);
    Matrix matP= eye(matK.rows, matK.cols);
    LUdecomp(matK, matL, matU,matP);
    solveLU(matL, matU, vecd, matX,matP);

    printMat(matX, "matX")
    return 0;
}
```

output

```
matX =
       1.159364
       3.366614
       4.347614
```

Warning

- L, U, P는 n x n 행렬이어야 한다.
- b, x는 n x 1 행렬이어야 한다.
- L은 아래삼각행렬, U는 위삼각행렬이다.

Error Handling

- L, U, P 행렬이 정방행렬이 아니면 오류가 발생한다.
- b, x 행렬의 사이즈는 같아야 한다.
- b행렬의 행의 개수는 L행렬의 행의 개수와 같아야 한다.
- b행렬의 열의 개수는 1이어야 한다.