Numerical Methods HW #03

1. 다음의 연립방정식을 Gauss 소거법을 사용하여 다음 물음과 같은 방법으로 풀어라.

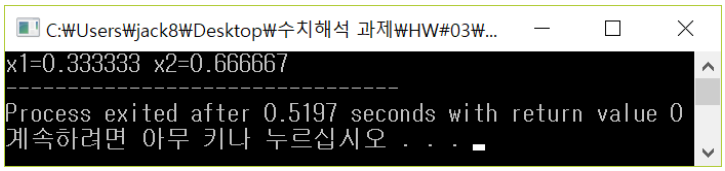
$$0.00005x_1 + 2x_2 = 1.33335$$
$$3x_1 + 18x_2 = 13$$

(a) 부분피봇팅을 실시하지 않고 풀어라.

<Code>

```c
#include<stdio.h>
int main()
{
	double A[2][3]={{0.00005,2,1.33335},{3,18,13}};
	double B[2][3];
	int i;
for(i=0;i<=2;i++)
	{
		B[0][i]=A[0][i]*A[1][0]/A[0][0];
	}
	for(i=0;i<=2;i++)
	{
		B[1][i]=B[0][i]-A[1][i];
	}
	double x1,x2;
	x2=B[1][2]/B[1][1];
	x1=(B[0][2]-x2*B[0][1])/B[0][0];
	printf("x1=%lf x2=%lf",x1,x2);
}
```

<Result>

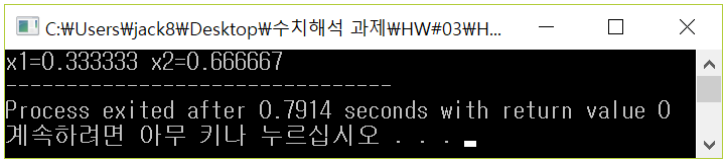


(b) 부분피봇팅을 실시하여 풀어라.

<Code>

```c
#include<stdio.h>
int main()
{
	double A[2][3]={{3,18,13},{0.00005,2,1.33335}};
	int i;
	double B[2][3];
	for(i=0;i<=2;i++)
	{
		B[0][i]=A[0][i]*A[1][0]/A[0][0];
	}
	for(i=0;i<=2;i++)
	{
		B[1][i]=B[0][i]-A[1][i];
	}
	double x1,x2;
	x2=B[1][2]/B[1][1];
	x1=(B[0][2]-x2*B[0][1])/B[0][0];
	printf("x1=%lf x2=%lf",x1,x2);
}
```

<Result>



2. Crout법을 이용하여 다음 행렬의 역행렬을 구하라.

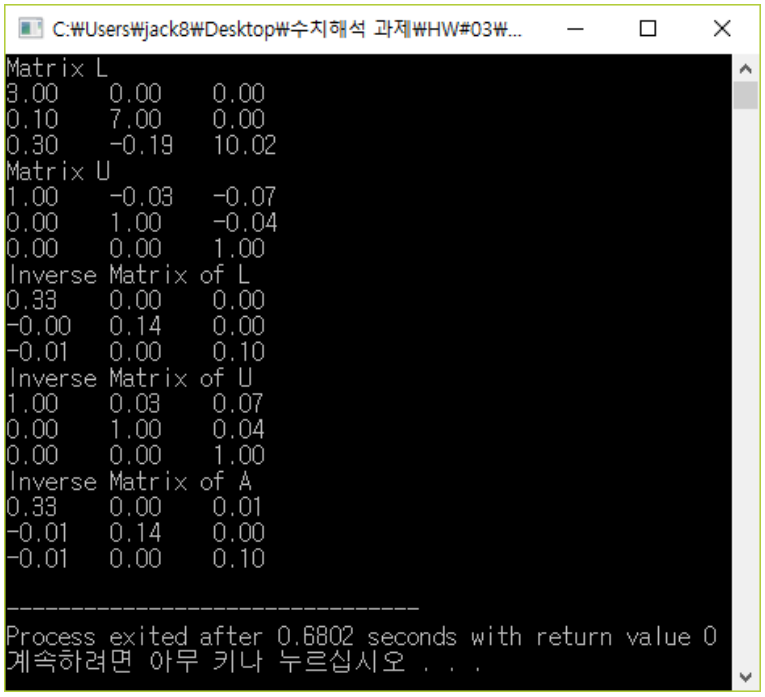
$$A = \begin{bmatrix} 3 & -0.1 & -0.2 \\ 0.1 & 7 & -0.3 \\ 0.3 & -0.2 & 10 \end{bmatrix}$$

<Code>

```
#include<stdio.h>
int main()
{
	double A[3][3]={{3,-0.1,-0.2},{0.1,7,-0.3},{0.3,-0.2,10}};
	double L[3][3]={0}, U[3][3]={0};
	int i,j;
	for(i=0;i<=2;i++)
	{
		L[i][0]=A[i][0];
		U[0][i]=A[0][i]/L[0][0];
	}
	L[1][1]=A[1][1]-L[1][0]*U[0][1];
	L[2][1]=A[2][1]-L[2][0]*U[0][1];
	L[2][2]=A[2][2]-L[2][0]*U[0][2]-L[2][1]*U[1][2];
	U[1][1]=1;	U[2][2]=1;
	U[1][2]=(A[1][2]-L[1][0]*U[0][2])/L[1][1];
	printf("Matrix L\n");
	for(j=0;j<=2;j++)
	{
		for(i=0;i<=2;i++)
		{
			printf("%.2lf	",L[j][i]);
		}
		printf("\n");
	}
	printf("Matrix U\n");
	for(j=0;j<=2;j++)
	{
		for(i=0;i<=2;i++)
		{
			printf("%.2lf	",U[j][i]);
		}
		printf("\n");
	}
	double invL[3][3]={{1,0,0},{0,1,0},{0,0,1}};
	double invU[3][3]={{1,0,0},{0,1,0},{0,0,1}};
	for(i=0;i<=2;i++)
	{
		invL[0][i]=invL[0][i]/L[0][0];
	}
	for(i=0;i<=2;i++)
	{
		invL[1][i]=invL[1][i]-invL[0][i]*L[1][0];
	}
	for(i=0;i<=2;i++)
	{
		invL[1][i]=invL[1][i]/L[1][1];
	}
	for(i=0;i<=2;i++)
	{
		invL[2][i]=invL[2][i]-invL[0][i]*L[2][0]-invL[1][i]*L[2][1];
	}
	for(i=0;i<=2;i++)
	{
		invL[2][i]=invL[2][i]/L[2][2];
	}
	for(i=0;i<=2;i++)
	{
		invU[1][i]=invU[1][i]-invU[2][i]*U[1][2];
	}
	for(i=0;i<=2;i++)
	{
		invU[0][i]=invU[0][i]-invU[1][i]*U[0][1]-invU[2][i]*U[0][2];
	}
```

```
        printf("Inverse Matrix of L\n");
        for(i=0;i<=2;i++)
        {
                for(j=0;j<=2;j++)
                {
                        printf("%.2lf     ",invL[i][j]);
                }
                printf("\n");
        }
        printf("Inverse Matrix of U\n");
        for(i=0;i<=2;i++)
        {
                for(j=0;j<=2;j++)
                {
                        printf("%.2lf     ",invU[i][j]);
                }
                printf("\n");
        }
        double invA[3][3]={0};
        int k;
        printf("Inverse Matrix of A\n");
        for(i=0;i<=2;i++)
        {
                for(j=0;j<=2;j++)
                {
                        for(k=0;k<=2;k++)
                        {
                                invA[i][j]=invA[i][j]+invU[i][k]*invL[k][j];
                        }
                        printf("%.2lf     ",invA[i][j]);
                }
                printf("\n");
        }
}
```

<Result>



3. 다음 연립방정식을 TDMA로 풀어라.

$$\begin{bmatrix} 3 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & 0 \\ 0 & -1 & 3 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & 0 \\ 0 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & -1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 100 \\ 150 \\ 150 \\ 150 \\ 150 \\ 100 \end{bmatrix}$$

<Code>

```
#include<stdio.h>
int main()
{
	double a=-1,b=3,c=-1;
	double d[6]={100,150,150,150,150,100};
	double p[6]={0}, q[6]={0};
	p[0]=-c/b; q[0]=d[0]/b;
	int i;
	for(i=1;i<=5;i++)
	{
		q[i]=(d[i]-a*q[i-1])/(a*p[i-1]+b);
		if(i==5) break;
		p[i]=-c/(a*p[i-1]+b);
	}
	double x[6];
	x[5]=q[5];
	for(i=4;i>=0;i--)
	{
		x[i]=p[i]*x[i+1]+q[i];
	}
	printf("x= ");
	for(i=0;i<=5;i++)
	{
		printf("%lf	",x[i]);
	}
	printf("\np= ");
	for(i=0;i<=5;i++)
	{
		printf("%lf	",p[i]);
	}
	printf("\nq= ");
	for(i=0;i<=5;i++)
	{
		printf("%lf	",q[i]);
	}
}
```

<Result>

```
C:₩Users₩jack8₩Desktop₩수치해석 과제₩HW#03₩HW#03_3.exe
x= 73.076923    119.230769      134.615385      134.615385      119.230769      73.076923
p= 0.333333     0.375000        0.380952        0.381818        0.381944        0.000000
q= 33.333333    68.750000       83.333333       89.090909       91.319444       73.076923
--------------------------------
Process exited after 0.7134 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

4. 다음 연립방정식을 Jacobi 반복법과 Gauss-Seidel 법으로 각각 풀어라. 단, 방정식의 해는 유효숫자 5자리까지 정확히 구하여라.

$$10x_1 - 3x_2 + 6x_3 = 24.5$$
$$x_1 + 8x_2 - 2x_3 = -10$$
$$-2x_1 + 4x_2 - 9x_3 = -50$$

(a) Jacobi 반복법

<Code>

```
#include<stdio.h>
#include<math.h>
int main()
{
	double a[3][3]={{10,-3,6},{1,8,-2},{-2,4,-9}};
	double b[3]={24.5,-10,-50};
	double x[3]={0};
	double X[3]={0};
	double error=0.5/pow(10,5);
	int i;
	for(i=1;i<=20;i++)
	{
		X[0]=(b[0]-a[0][1]*x[1]-a[0][2]*x[2])/a[0][0];
		X[1]=(b[1]-a[1][0]*x[0]-a[1][2]*x[2])/a[1][1];
		X[2]=(b[2]-a[2][0]*x[0]-a[2][1]*x[1])/a[2][2];
		printf("%d x1=%lf x2=%lf   x3=%lf\n",i,X[0],X[1],X[2]);
		double e1,e2,e3;
		e1=fabs((X[0]-x[0])/X[0]);
		e2=fabs((X[1]-x[1])/X[1]);
		e3=fabs((X[2]-x[2])/X[2]);
		printf("error1=%lferror2=%lf        error3=%lf\n",e1,e2,e3);
		if(i!=1)
		{
			if(e1<error&&e2<error&&e3<error) break;
		}
		x[0]=X[0]; x[1]=X[1]; x[2]=X[2];
	}
}
```

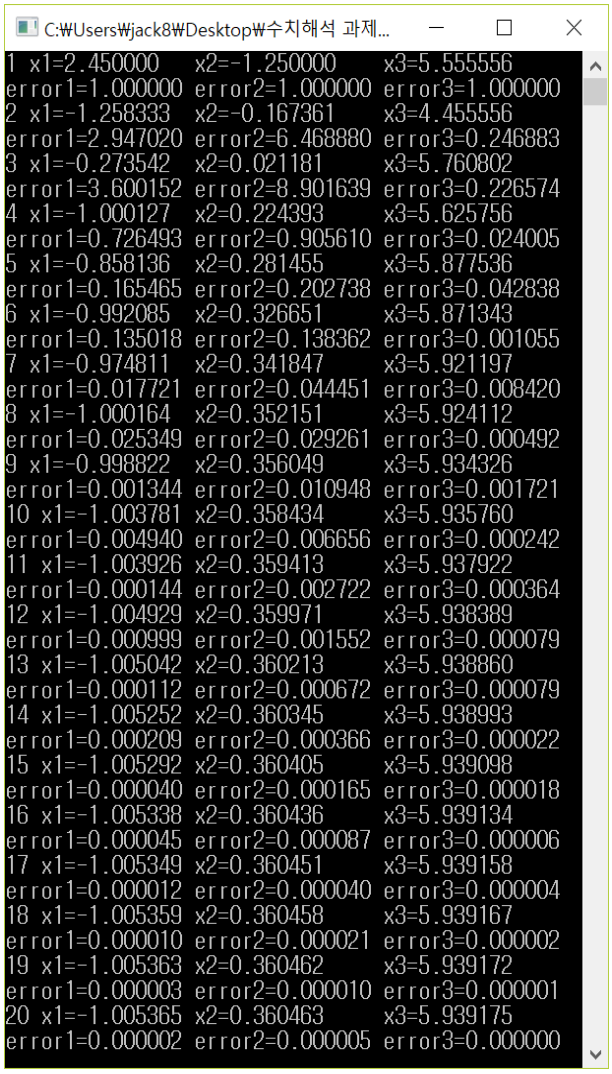
(b) Gauss-Seidel 법

<Code>

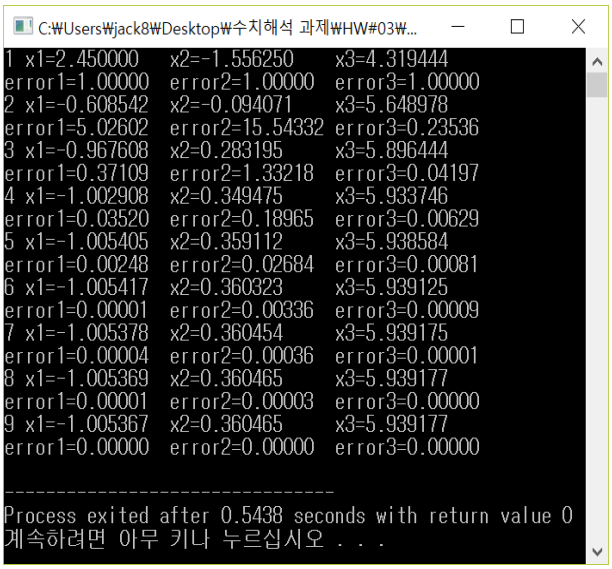
```
#include<stdio.h>
#include<math.h>
int main()
{
	double a[3][3]={{10,-3,6},{1,8,-2},{-2,4,-9}};
	double b[3]={24.5,-10,-50};
	double x[3]={0};
	double X[3]={0};
	double error=0.5/pow(10,5);
	int i;
	for(i=1;i<=20;i++)
	{
		X[0]=(b[0]-a[0][1]*x[1]-a[0][2]*x[2])/a[0][0];
		X[1]=(b[1]-a[1][0]*X[0]-a[1][2]*x[2])/a[1][1];
		X[2]=(b[2]-a[2][0]*X[0]-a[2][1]*X[1])/a[2][2];
		printf("%d x1=%lf x2=%lf   x3=%lf\n",i,X[0],X[1],X[2]);
		double e1,e2,e3;
		e1=fabs((X[0]-x[0])/X[0]);
		e2=fabs((X[1]-x[1])/X[1]);
		e3=fabs((X[2]-x[2])/X[2]);
		printf("error1=%lferror2=%lf        error3=%lf\n",e1,e2,e3);
		if(i!=1)
		{
			if(e1<error&&e2<error&&e3<error) break;
		}
		x[0]=X[0]; x[1]=X[1]; x[2]=X[2];
	}
}
```

<Result>

(a) Jacobi 반복법



(b) Gauss-Seidel 법



5. 다음 연립방정식을 주어진 이완계수 w을 갖고 SOR 또는 SUR로 풀어라. 그리고, Gauss-Seidel 법으로 풀어 반복횟수를 비교하여라. 단, 방정식의 해는 유효숫자 5자리 이내에 있도록 하라.

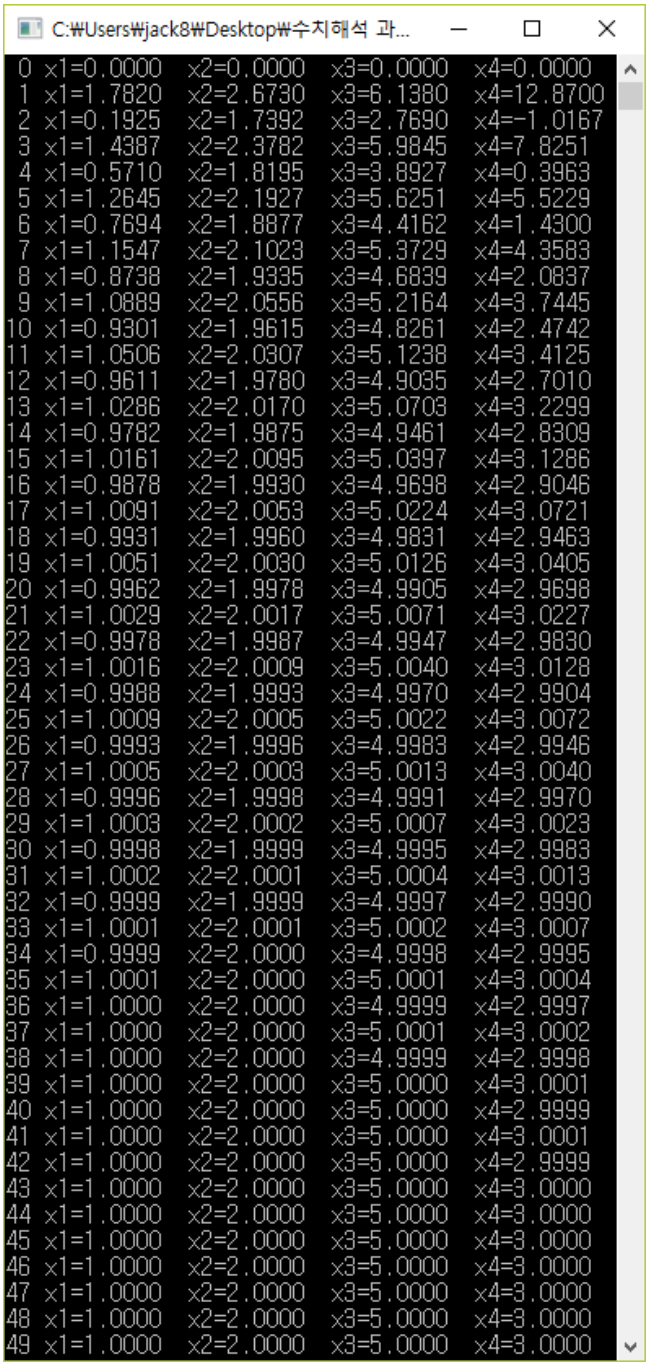
$$2x_1 + 10x_2 + x_3 = 27$$
$$x_1 + x_2 + 5x_3 + x_4 = 31$$
$$3x_1 + x_2 + x_3 + x_4 = 13$$
$$15x_1 + 2x_2 + x_3 + x_4 = 27$$
$$w = 0.99$$

<Code>

```
#include<stdio.h>
#include<math.h>
int main()
{
	double a[4][4]={{15,2,1,1},{2,10,1,0},{1,1,5,1},{3,1,1,1}};
	double b[4]={27,27,31,13};
	double w=0.99; //or 1
	double x[4]={0,0,0,0};
	double X[4]={0,};
	double error=0.5/pow(10,5);
	int i,j,k;
	for(i=1;i<=200;i++)
	{
		printf("%2d x1=%.4lf  x2=%.4lf  x3=%.4lf  x4=%.4lf\n",i-1,x[0],x[1],x[2],x[3]);
		for(j=0;j<=3;j++)
		{
			X[j]=x[j]+w*b[j]/a[j][j];
			for(k=0;k<=j-1;k++)
			{
				X[j]=X[j]-w*a[j][k]*x[k]/a[j][j];
			}
			for(k=j;k<=3;k++)
			{
				X[j]=X[j]-w*a[j][k]*x[k]/a[j][j];
			}
		}
		double e1,e2,e3,e4;
		e1=fabs((X[0]-x[0])/X[0]);
		e2=fabs((X[1]-x[1])/X[1]);
		e3=fabs((X[2]-x[2])/X[2]);
		e4=fabs((X[3]-x[3])/X[3]);
		if(i!=1)
		{
			if(e1<error&&e2<error&&e3<error&&e4<error) break;
		}
		x[0]=X[0]; x[1]=X[1]; x[2]=X[2]; x[3]=X[3];
	}
}
```

<Result>

(a) SUR(w=0.99<1)



(b) Gauss-Seidel 법(w=1)