```
#include <iostream>

#include<stdlib.h>

#include<time.h>

using namespace std;



//Funciones a utilizar

bool ComprobarColumna(float [][100], int);//Se comprueba que el sistema Ax = B se pueda resolver

void ArreglandoMatriz(float [][100], int, float [100]);//Esta funcion quita el CERO de la posicion [0,0] en los casos que se den.

void CambiarFila(float matrizOriginal[][100], float matrizBOriginal[], int filaActual, int dimensionDeMatriz);//Esta se usa si algun elemento de la diagonal es CERO


//Funciones de operaciones fijas. Son funciones que se tienen que usar si o si para que se el resultado final

void HacerCeroPrimeraColumna(float matrizOriginal[][100], float matrizBOriginal[], int dimensionDeMatriz);

void DiagonalEnUno(int dimensioDeMatriz, float matrizOriginal[][100], float matrizBOriginal[]);

void HacerCeroDebajoDeDiagonal(float matrizOriginal[][100], float matrizBOriginal[], int dimensionDeMatriz);

void CrearMatrizIdentidad(float matrizOriginal[][100], float matrizBOriginal[], int dimensionDeMatriz);



int main()

{

   srand(time(NULL));

   int dimensionN = 4 ;


   int Aleatorio = -1;
```

```
   cout<<"Se va a intentar calcular el sistme Ax = B. Para eso debe ingresar los datos de A y B"<<endl;

   do

   {

      cout<<"\nIngrese el la dimension de la Matriz A (n x n) y el cual tambien sera el de la matriz columna B (n)."<<endl;

      cout<<"n ----> ";

      cin>>dimensionN;

   }

   while(dimensionN > 99 || dimensionN < 1);

   //Arreglos A utilizar

   float matrizA[dimensionN][dimensionN],vectorX[dimensionN],matrizColumnaB[dimensionN], MatrizAcomprobar[100][100];

   float MatrizColumnaBComprobar[100];



   // El siguiente DO{}WHILE() da la opcion de crear automaticamente la MATRIZ A y el VECTOR B de una dimension n

   // aleatoriamente. Se puede ingresar el valor uno por uno si se escoge la opcion 0 ( cero ).

   do

   {

      cout<<"\nDesea crear la matriz A y la matriz Columna B de forma Aleatoria y Automatica? : \n1 = SI \n0 = NO\n";

      cout<<"\n----> ";

      cin>> Aleatorio;


      if(Aleatorio > 1 || Aleatorio < 0)

      {

         cout<<"\nHa ingresado un valor no permitido. Intente de nuevo\n"<<endl;

      }
```

```
  }
  while(Aleatorio > 1 || Aleatorio < 0);

  if(Aleatorio == 1)//Se ejecutara automaticamente la asignacion de valores en la Matriz A y en la matriz Columna B
  {
    cout<<"Creando la Matriz A y la Matriz Columna B aleatoriamente"<<endl;
    //Los siguientes Siclos FOR son paras rellenar la Matriz A del sistema Ax = B
    for(int i = 0; i< dimensionN; i++)
    {
      for(int j = 0; j < dimensionN; j++)
      {
        matrizA[i][j] = 1 + rand()%10;
        MatrizAcomprobar[i][j] = matrizA[i][j];
      }
      matrizColumnaB[i] = 1 + rand()%10;
      MatrizColumnaBComprobar[i] = matrizColumnaB[i];
    }
  }
  else
  {
    //Los siguientes Siclos For son paras rellenar la Matriz A de sistema Ax = B
    for(int i = 0; i< dimensionN; i++)
    {
      for(int j = 0; j < dimensionN; j++)
      {
        cout<<"Ingrese un valor al elemento A["<<(i+1)<<"]["<<(j+1)<<"] ---> ";
        cin>>matrizA[i][j];
        MatrizAcomprobar[i][j] = matrizA[i][j];
```

```
            cout<<endl;
        }
        cout<<"Ingrese el valor al elemento B["<<(i+1)<<"] ---> ";
        cin>>matrizColumnaB[i];
        MatrizColumnaBComprobar[i] = matrizColumnaB[i];
    }
}

cout<<"\nLa Matriz A Formada es: \n"<<endl;
// Los siguientes siclos FOR son para mostrar como quedo la matriz creada anteriormente.
for(int i = 0; i < dimensionN; i++)
{
    for(int j = 0; j < dimensionN; j++)
    {
        // Los bloques IF son para imprimir en orden la matriz formada
        if(matrizA[i][j] < 10 && matrizA[i][j] > -10)
        {
            cout<<"| "<<matrizA[i][j];
        }
        if(matrizA[i][j] >= 10 || matrizA[i][j] <= -10)
        {
            cout<<"|"<<matrizA[i][j];
        }
    }
    cout<<" |"<<endl;
}

cout<<"\nLa Matriz Columna B Formada es: \n";
```

```
   for(int x  = 0; x < dimensionN; x++)
   {
      cout<<"|"<<matrizColumnaB[x]<<"|"<<endl;


   }


   //La funcion COMPROBARCOLUMNA verifica si alguna columna esta hecha completamente de CEROS (0)
   // y si es el caso, el sistema no tiene solucion.
   if(ComprobarColumna(MatrizAcomprobar, dimensionN))
   {
      if(MatrizAcomprobar[0][0]==0)
      {
         //Si el primer elemento es CERO no se puede realizar los calculos correctamsnte, por eso se cambiara
         cout<<"\nEl primer elemento es un CERO (0) se cambiara toda la primera fila por otra que su primer elemento no sea cero."<<endl;
         ArreglandoMatriz(MatrizAcomprobar,dimensionN, MatrizColumnaBComprobar);
      }
      HacerCeroPrimeraColumna(MatrizAcomprobar, MatrizColumnaBComprobar, dimensionN);
      HacerCeroDebajoDeDiagonal(MatrizAcomprobar, MatrizColumnaBComprobar, dimensionN);
      CrearMatrizIdentidad(MatrizAcomprobar, MatrizColumnaBComprobar, dimensionN);


      //Resigna los valores de las matrices que se usaron para realizar las operaciones a las matrices originales
      for(int i = 0; i < dimensionN; i++)
      {
         for(int j = 0; j < dimensionN; j++)
         {
            matrizA[i][j] = MatrizAcomprobar[i][j];
```

```
    }
    vectorX[i] = MatrizColumnaBComprobar[i];
}

//El siguiente bloque es para visualizar como cambia la matriz al final de todo el proceso
cout<<"\nLa Matriz A Formada (hecha Matriz IDENTIDAD) es: \n"<<endl;
// Los siguientes siclos FOR son para mostrar como quedo la matriz creada anteriormente.
for(int i = 0; i < dimensionN; i++)
{
    for(int j = 0; j < dimensionN; j++)
    {
        //Esta bloque IF evita que exitan CEROS (0) con signo negativo (-0)
        if(matrizA[i][j] == 0)
        {
            matrizA[i][j] *= matrizA[i][j];
        }

        cout<<"| "<<matrizA[i][j];

    }
    cout<<" |"<<endl;
}

cout<<"\nLos valores de las incognitas de la matriz columna X es: \n";

for(int x  = 0; x < dimensionN; x++)
{
    if(vectorX[x] == 0)
    {
```

```
            vectorX[x] *= vectorX[x];
        }
        cout<<"El elemento X["<<(x+1)<<"] es: ";
        cout<<"|"<<vectorX[x]<<"|"<<endl;

    }


  }
  else
  {
    cout<<"\nEl sistema no tienen solucion. No se haran calculos"<<endl;
  }


  cout << "**********Fin del
programa**********" << endl;
  return 0;
}

void CrearMatrizIdentidad(float matrizOriginal[][100], float matrizBOriginal[], int
dimensionDeMatriz)
{
  int filaACambiar = dimensionDeMatriz-2, inicioVertical = 1, inicioHorizontal =
dimensionDeMatriz-1;
  float valorAMultiplicar = matrizOriginal[filaACambiar][inicioHorizontal];
  int elementoBAnterior = dimensionDeMatriz-1, elementoAEliminarB = dimensionDeMatriz-2;

  for(int columna = inicioHorizontal; columna > 0; columna-- )
  {
```

```
      for(int fila = filaACambiar; fila >=0; fila--)
      {
         //Esto hace el proceso de Convertir en CERO la parte superior de la Diagonal
         valorAMultiplicar = (matrizOriginal[fila][columna] * -1);
         matrizOriginal[fila][columna] += valorAMultiplicar;

         //Esto realiza el proceceso en la matriz columna B
         elementoBAnterior = fila+1;
         valorAMultiplicar *= matrizBOriginal[elementoBAnterior];

         matrizBOriginal[fila] += valorAMultiplicar;
      }

      elementoBAnterior--;

      filaACambiar--;
   }
}



void HacerCeroDebajoDeDiagonal(float matrizOriginal[][100], float matrizBOriginal[], int
dimensionDeMatriz)
{
   DiagonalEnUno(dimensionDeMatriz,matrizOriginal,matrizBOriginal);
   int filaACambiar = 2, inicioVertical = 2, filaAnterior = 1, inicioHorizontal = 1;
   float valorAMultiplicar = matrizOriginal[filaACambiar][inicioHorizontal];
   int elementoBAnterior = 1, elementoAEliminarB = 2, elementoFijo = 0;
```

```
   do
   {
      for(int fila = inicioVertical; fila< dimensionDeMatriz; fila++)
      {
         valorAMultiplicar = matrizOriginal[fila][inicioHorizontal] * -1;
         //Procedimiento para operar en la matriz columna B
         matrizBOriginal[elementoAEliminarB] +=  matrizBOriginal[elementoBAnterior] *
valorAMultiplicar;

         //Estas variables controlan que la interaccion con la matriz columna B
         elementoBAnterior++;
         elementoAEliminarB++;


         for(int elemento = inicioHorizontal; elemento < dimensionDeMatriz; elemento++)
         {
            valorAMultiplicar *= (matrizOriginal[filaAnterior][elemento]);
            matrizOriginal[fila][elemento] += valorAMultiplicar;
         }

      }

      //Estas variables controlan el recorrido por debajo de la Diagonal
      filaAnterior++;//Esta inicia en 1 (uno)
      inicioHorizontal++;// Esta inicia en 1 (uno)

      //Esta es la varible bandera. Cuando esta sea igual a las dimensiones de la matriz, se saldra del
ciclo
      inicioVertical++;
```

```
      //En cada ejecucion se vuelven a colocar la diagonal en 1 para tener un elemento pivote siempre.
      DiagonalEnUno(dimensionDeMatriz,matrizOriginal,matrizBOriginal);
   }
   while(inicioVertical < dimensionDeMatriz);



}
void HacerCeroPrimeraColumna(float matrizOriginal[][100], float matrizBOriginal[], int dimensionDeMatriz)
{
   int filaAModificar = 1, primeraFila = 0;
   float valorAMultiplacar = 1, primerElemento = matrizOriginal[0][0];

   //Este siclo for hace que el primer elemento [0,0] se convierta en UNO (1) como elemento pivote
   for(int elemento = 0; elemento < dimensionDeMatriz; elemento++)
   {
      matrizOriginal[0][elemento] = (matrizOriginal[0][elemento]/primerElemento);
   }
   matrizBOriginal[0] = (matrizBOriginal[0]/primerElemento);


   //Los siguientes siclos FOR hacen la operacion de crear la primera columna en ceros. exceptuando el primer elemento
   for(int i = filaAModificar; i < dimensionDeMatriz; i++)
   {

      valorAMultiplacar = matrizOriginal[i][0]; // V.M. = 5
```

```
      //Esto modifica la matriz original en su totalidad
      for(int j = 0; j < dimensionDeMatriz; j++)
      {
         valorAMultiplacar = (matrizOriginal[primeraFila][j] * valorAMultiplacar) * -1; // V.M. =
(1*5)*-1 = -5
         matrizOriginal[i][j] += valorAMultiplacar; // M.O. = 5 + (-5) = 0
      }
      //Se realizan con la matriz columna B
      matrizBOriginal[i] += (matrizBOriginal[primeraFila] * valorAMultiplacar) * -1;
      //matrizBOriginal[i] += valorAMultiplacar;

   }
}


void DiagonalEnUno(int dimensioDeMatriz, float matrizOriginal[][100], float matrizBOriginal[])
{
   float diagonal = 1;
   for (int D=1; D<dimensioDeMatriz; D++)
   {
      diagonal = matrizOriginal[D][D];
      if(diagonal == 0)
      {
         CambiarFila(matrizOriginal, matrizBOriginal, D, dimensioDeMatriz);
      }
      for (int M=1; M<dimensioDeMatriz; M++)
      {
         matrizOriginal[D][M] = (matrizOriginal[D][M])/(diagonal);
      }
```

```
      matrizBOriginal[D]= matrizBOriginal[D]/(diagonal);


   }
}

void CambiarFila(float matrizOriginal[][100], float matrizBOriginal[], int filaActual, int
dimensionDeMatriz)
{
   float filaACambiar[100], filaSeleccionadaAUtilizar[100];
   int filaElegida = filaActual + 1; //Esta variable seleccionada servira para encontrar una fila que no
tenga un cero debajo de la diagonal


   //Se hace una copia de la fila que se cambiara a partir de la DIAGONAL
   for(int i = 0; i < dimensionDeMatriz; i++)
   {
      filaACambiar[i] = matrizOriginal[filaActual][i];
   }


   do
   {


      if(matrizOriginal[filaElegida][filaActual] == 0)
      {
         filaElegida++;
      }


   }
   while(matrizOriginal[filaElegida][filaActual] == 0 || filaElegida < dimensionDeMatriz); //En este
caso la varible FILAACTUAL sirve como selector de columna
```

```
   //Se hace una copia de la fila que sera cambia por la anterior copia hecha de la fila superior
   for(int j = 0; j < dimensionDeMatriz; j++)
   {
      filaSeleccionadaAUtilizar[j] = matrizOriginal[filaElegida][j];
   }
   //Se permutan las dos filas para que en la DIAGONAL no haya un CERO (0) ejemplo: R3 ---> R5
   for(int m = 0; m < dimensionDeMatriz; m++)
   {
      matrizOriginal[filaActual][m] = filaSeleccionadaAUtilizar[m];
      matrizOriginal[filaElegida][m] = filaACambiar[m];

   }
}

void ArreglandoMatriz(float matrizOriginal[][100], int dimensionDeMatriz, float
MatrizColumnaB[100])
{
   float matrizTemporal[100], matrizBTemporal[3];
   int filaSeleccionada = 1, elementoNoCero = 0;

   do
   {
      //Busca un el primer elemento no CERO en la primero columna
      if(matrizOriginal[filaSeleccionada][0] == 0)
      {
         filaSeleccionada++;
      }
   }
   while(matrizOriginal[filaSeleccionada][0] == 0);
```

```
   //En el siguiente ciclo se ordenan los elementos de las 2 filas que seran intercabiadas,
   for (int k = 0; k < dimensionDeMatriz; k++)
   {
      //Resumen de lo que se hace aqui: A <- B, B <- C, C <- A
      matrizTemporal[k] = matrizOriginal[filaSeleccionada][k];
      matrizOriginal[filaSeleccionada][k] = matrizOriginal[0][k];
      matrizOriginal[0][k] = matrizTemporal[k];
   }

   //Procedimiento para la matriz columna B
   matrizBTemporal[0] = MatrizColumnaB[filaSeleccionada];
   MatrizColumnaB[filaSeleccionada] = MatrizColumnaB[0];
   MatrizColumnaB[0] = matrizBTemporal[0];
}

bool ComprobarColumna(float matrizOriginal[][100], int dimensioDeMatriz)
{
   int ColumnaAVerificar = 0, ceroEncontrado = 0;
   int ceroDebajoDeDiagonal = 0, abajoDediagonal = 0, bajarFila = 0;//Estas variables sirven para hacer contar los CEROS que hay debajo
   // de cada numero de la diagonal. En caso de solo sean CEROS debajo de esos numeros. el sistemo no tiene solucion

   bool columnaEnCero = false;

   for(int fila = 0; fila<dimensioDeMatriz; fila++)
   {
      if(bajarFila < dimensioDeMatriz-1)
      {
```

```
        abajoDediagonal = matrizOriginal[fila + bajarFila][fila];
        if(abajoDediagonal == 0)
        {
            ceroDebajoDeDiagonal++;
            bajarFila++;
        }
        else
        {
            ceroDebajoDeDiagonal = 0;
            bajarFila = 0;
        }

    }
    for(int columna = 0; columna < dimensioDeMatriz; columna++)
    {

        if(matrizOriginal[fila][columna] == 0)
        {
            ceroEncontrado++ ;
        }
        else
        {
            ceroEncontrado = 0;
            break;
        }
        if((ceroEncontrado  >= dimensioDeMatriz) || (ceroDebajoDeDiagonal >= dimensioDeMatriz
- bajarFila))
        {
            columnaEnCero = true;
```

```
        }
      }
   }

   //se verifica si alguna columna es llena de CERO, en caso que si lo sea retorna FALSE, para indicar que no tienen solucion el problema
   //En caso que ninguna columna este llena de CERO, el sistema tiene solucion y continua las operaciones.
   if(columnaEnCero)
   {
      return false;
   }
   else
   {
      return true;
   }

}
```