PRACTICA 3 .NET

2) Implementar un método para imprimir por consola todos los elementos de una matriz (arreglo de dos dimensiones) pasada como parámetro. Debe imprimir todos los elementos de una fila en la misma línea en la consola.

```
void ImprimirMatriz(double[,] matriz)
```

Ayuda: Si `A` es un arreglo, `A.GetLength(i)` devuelve la longitud del arreglo en la dimensión `i`.

```
double [,] matriz = new double [4,4];

for (int i=0;i<16;i++){
    matriz[i/4,i%4] =i +0.5;
}
ImprimirMatriz(matriz);

void ImprimirMatriz(double[,] matriz){
    for (int i=0;i<16;i++){
        Console.Write(matriz[i/4,i%4]+ "|| ");
        Console.Write((i%4)==3 ? "\n": " ");  //para imprimir por fila//
    }

}
```

3) Implementar el método **ImprimirMatrizConFormato**, similar al anterior pero ahora con un parámetro más que representa la plantilla de formato que debe aplicarse a los números al imprimirse. La plantilla de formato es un string de acuerdo a las convenciones de formato compuesto, por ejemplo **"0.0"** implica escribir los valores reales con un dígito para la parte decimal.

```
void ImprimirMatrizConFormato(double[,] matriz, string formatString)
```

```
Console.WriteLine("INGRESE EL FORMATO DE LA MATRIZ");
String? st = Console.ReadLine();
double [,] matriz = new double [4,4];
for (int i=0;i<16;i++){
    matriz[i/4,i%4] = i;
}
ImprimirMatrizConFormato(matriz,st);

void ImprimirMatrizConFormato (in double[,] matriz,String? Format){
    for (int i=0;i<16;i++){
        Console.Write(matriz[i/4,i%4].ToString(Format));
        Console.Write((i%4)==3 ? "\n": " ");
```

```
    }
}
```

4) Implementar los métodos **GetDiagonalPrincipal** y **GetDiagonalSecundaria** que devuelven un vector con la diagonal correspondiente de la matriz pasada como parámetro. Si la matriz no es cuadrada generar una excepción **ArgumentException**.

```
double[] GetDiagonalPrincipal(double[,] matriz)
double[] GetDiagonalSecundaria(double[,] matriz)
```

```
Console.WriteLine("INGRESE EL FORMATO DE LA MATRIZ");
String? st = Console.ReadLine();
double [,] matriz = new double [4,4];
for (int i=0;i<16;i++){
    matriz[i/4,i%4] = i+0.5;
}
double [ ]v = GetDiagonalPrincipal(matriz);
foreach (double i in v)
Console.WriteLine(i);
Console.WriteLine("------------------------");
Console.WriteLine("-------------Vector invertido-------------");
double [ ]v2 = GetDiagonalSecundaria(matriz);
foreach (double i in v2)
Console.WriteLine(i);


double [] GetDiagonalPrincipal (in double [,] matriz){
    if (matriz.GetLength(0)!= matriz.GetLength(1)){
        throw new ArgumentException ("LA MATRIZ NO ES CUADRADA");
    }
    double []DiagonalPrincipal = new double[4];
    {
        for (int i=0;i<4;i++){
        DiagonalPrincipal[i]=matriz[i,i];
    }
    }
    return DiagonalPrincipal;
}



double [] GetDiagonalSecundaria(in double [,] matriz){
    if (matriz.GetLength(0)!= matriz.GetLength(1)){
        throw new ArgumentException ("LA MATRIZ NO ES CUADRADA");
    }
    double []DiagonalSecundaria = new double[4];
```

```
    {
        int j=3;
        for (int i=0;i<4;i++){
        DiagonalSecundaria[i]=matriz[j--,i];
    }
    }
    return DiagonalSecundaria;
}
```

6) Implementar los siguientes métodos que devuelvan la suma, resta y multiplicación de matrices pasadas como parámetros. Para el caso de la suma y la resta, las matrices deben ser del mismo tamaño, en caso de no serlo devolver **`null`**. Para el caso de la multiplicación la cantidad de columnas de **`A`** debe ser igual a la cantidad de filas de **`B`**, en caso contrario generar una excepción **`ArgumentException`**.

```
double[,]? Suma(double[,] A, double[,] B)
double[,]? Resta(double[,] A, double[,] B)
double[,] Multiplicacion(double[,] A, double[,] B)
```

```
double [,] m = new double [,] {{1,2},{4,5}};
double [,] m2 = new double [,] {{1,2,3},{4,5,6}};

double[,] sumatoria;
sumatoria = Multiplicacion(m,m2);

if (sumatoria != null){
for (int i=0;i<sumatoria.GetLength(0);i++){
    for (int j=0;j<sumatoria.GetLength(1);j++)
        Console.WriteLine(sumatoria[i,j]);
     }
}
else
    Console.WriteLine("SUMATORIA ES NULL");



double[,]? Suma(double[,] A, double[,] B){

   double [,]? sumas = null;
    if (sonIguales(A,B)){ //si las filas y columnas son iguales
        sumas = new double [A.GetLength(0),A.GetLength(1)] ;
        int Columnas = A.GetLength(0);
       for (int i=0;i<sumas.GetLength(0);i++){
    for (int j=0;j<sumas.GetLength(1);j++)
        sumas[i,j]  = A[i,j] + B[i,j];
    }

    }
    return sumas;
}


double[,]? Resta(double[,] A, double[,] B){
    double [,] ? Resta = null;
    if (sonIguales(A,B)){
        Resta = new double [A.GetLength(0),A.GetLength(1)] ;
          int Columnas = A.GetLength(0);
        for (int i=0;i<Columnas;i++)
            Resta [i/Columnas,i%Columnas] = A [i/Columnas,i%Columnas] -
B[i/Columnas,i%Columnas];
    }
    return Resta;
}
```

```
 Boolean sonIguales(double[,]A,double[,]B){
    return A.GetLength(0)== B.GetLength(0) && (A.GetLength(1)==B.GetLength(1));
    }


    double[,] Multiplicacion(double[,] A, double[,] B){
      {
        if (A.GetLength(1)! = B.GetLength(0))
            throw new ArgumentException("El número de columnas de A debe
coincidir con el número de filas de B.");

        double[,] Resultado = new double[A.GetLength(0), B.GetLength(1)];

        for (int i = 0; i < A.GetLength(0); i++)
        {
            for (int j = 0; j < B.GetLength(1); j++)
            {
                Resultado[i, j] = 0;
                for (int k = 0; k < A.GetLength(1); k++)
                {
                    Resultado[i, j] += A[i, k] * B[k, j];
                }
            }
        }

        return Resultado;
    }
}
```

7) ¿De qué tipo quedan definidas las variables **x**, **y**, **z** en el siguiente código?

```
int i = 10;
var x = i * 1.0;
var y = 1f;
var z = i * y;
```

Int i = 10 tipo integer;

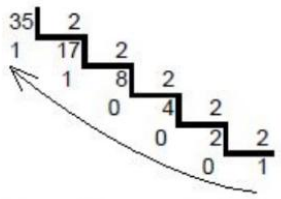
Var x=i * 1.0 Integer

Otras dos las toma como float.

12) Realizar un análisis sintáctico para determinar si los paréntesis en una expresión aritmética están bien balanceados. Verificar que por cada paréntesis de apertura exista uno de cierre en la cadena de entrada. Utilizar una pila para resolverlo. Los paréntesis de apertura de la entrada se almacenan en una pila hasta encontrar uno de cierre, realizándose entonces la extracción del último paréntesis de apertura almacenado. Si durante el proceso se lee un paréntesis de cierre y la pila está vacía, entonces la cadena es incorrecta. Al finalizar el análisis, la pila debe quedar vacía para que la cadena leída sea aceptada, de lo contrario la misma no es válida.

ESTA EN MI CELU XD

13) Utilizar la clase **Stack<T>** (pila) para implementar un programa que pase un número en base 10 a otra base realizando divisiones sucesivas. Por ejemplo para pasar 35 en base 10 a binario dividimos sucesivamente por dos hasta encontrar un cociente menor que el divisor, luego el resultado se obtiene leyendo de abajo hacia arriba el cociente de la última división seguida por todos los restos.



El resultado es 100011

```
Console.WriteLine("INGRESE UN NUMERO");

int n =int.Parse(Console.ReadLine()); //lo casteo
Stack<int>pila =new Stack<int>(); //genero la pila

while (n>0){
    pila.Push(n%2);
    n=n/2;
}
int binario=0;
while (pila.Count != 0 ){
    binario =pila.Pop()+binario*10 ;
}
Console.WriteLine(binario);
```

14) Utilizar la clase **Queue<T>** para implementar un programa que realice el cifrado de un texto aplicando la técnica de clave repetitiva. La técnica de clave repetitiva consiste en desplazar un carácter una cantidad constante de acuerdo a la lista de valores que se encuentra en la clave. Por ejemplo: para la siguiente tabla de caracteres:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z	sp
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28

Si la clave es 5,3,9,7 y el mensaje a cifrar es **"HOLA MUNDO"**, se cifra de la siguiente manera:

H	O	L	A	sp	M	U	N	D	O	← Mensaje original
8	16	12	1	28	13	22	14	4	16	← Código sin cifrar
5	3	9	7	5	3	9	7	5	3	← Clave repetida
13	19	21	8	5	16	3	21	9	19	← Código cifrado
M	R	T	H	E	O	C	T	I	R	← Mensaje cifrado

A cada carácter del mensaje original se le suma la cantidad indicada en la clave. En el caso que la suma fuese mayor que 28 se debe volver a contar desde el principio, Implementar una cola con los números de la clave encolados y a medida que se desencolen para utilizarlos en el cifrado, se vuelvan a encolar para su posterior utilización. Programar un método para cifrar y otro para descifrar.

```
Codificar(st,clave);




void Codificar(String st,int [] clave){
    Queue <int> q = new Queue <int>(clave);
    foreach (Char Carac in st){
        int aux;
        if (Carac =='Ñ')
            aux=15;
        else
            if(Carac==' ')
                aux= 28;
            else
                if (Carac <='N')
                    aux=Carac-'A'+1;
                else
```

```
                    aux=Carac-'A'+2;
        int queue = q.Dequeue();
        q.Enqueue(queue);
        aux+=queue;
        if (aux>28)
            aux=aux-28;
        // debo pasar el aux a chad
        Console.Write(ConvertirNumero(aux));
        //hasta ahi esta codificado en numeros
        //para imprimirlo
    }
}

Char ConvertirNumero(int aux){
if (aux==15) return 'Ñ';
if (aux==28) return ' ';
if (aux <=14)
     aux=aux+'A'-1;
else
    aux=aux+'A'-2;
return (char)aux;
}
```

15) ¿Qué salida por la consola produce el siguiente código?

```
int x = 0;
try
{
    Console.WriteLine(1.0 / x);
    Console.WriteLine(1 / x);
    Console.WriteLine("todo OK");
}
catch (Exception e)
{
    Console.WriteLine(e.Message);
}
```

¿Qué se puede inferir respecto de la excepción división por cero en relación al tipo de los operandos?

```
∞
Attempted to divide by zero.
```

16) Implementar un programa que permita al usuario ingresar números por la consola. Debe ingresarse un número por línea finalizado el proceso cuando el usuario ingresa una línea vacía. A medida que se van ingresando los valores el sistema debe mostrar por la consola la suma acumulada de los mismos. Utilizar **`double.Parse()`** y **`try/catch`** para validar que la entrada ingresada sea un número válido, en caso contrario advertir con un mensaje al usuario y proseguir con el ingreso de datos.

```
double suma = 0;
String num ;
while (true){
   Console.WriteLine("INGRESE NUMERO");
   num = Console.ReadLine();

   if (String.IsNullOrEmpty(num)){
        break;
   }
   //si no es vacio convierto el numero
   try{
        double aux = double.Parse(num);
        suma +=aux;
        Console.WriteLine("SUMA ACUMULADA :" + suma);
   }
   catch (FormatException)
   {
    Console.WriteLine("INGRESE UN NUMERO VALIDO");
   }
}
```

```
Bloque finally en Metodo1

Método 1 propagó una excepción no tratada

Método 2 propagó una excepción no tratada

Excepción InvalidCast en Metodo3

Método 3 propagó una excepción
```