****
// Problema nº 11: ¿v[i] = i?, v estrictamente creciente
****

Disponemos de un vector A de *N* elementos enteros ordenados en sentido creciente estricto, i.e., no hay elementos repetidos. Se pretende encontrar una posición I, tal que A[I] = I. Diseñe un algoritmo "Divide y Vencerás" que devuelva dicha posición o retorne el valor 0 cuando no exista ninguna.

```
int Decide (int v[], int li, int ls) {

   if (li > ls) // no existe
      return -1;
   else {
      int md = (li + ls)/2;
      if (v[md] == md) return md;
      else if (v[md] < md) return Decide (v,md+1,ls);
           else return Decide (v,li,md-1);
   }
}
```

****
// Problema nº 13: ¿v[i] = i?, v decreciente, no necesariamente estricto
****

Disponemos de un vector A de *N* enteros ordenados en sentido decreciente, no necesariamente estricto. Construya un procedimiento "Divide y Vencerás" tal que si existe una posición I, tal que A[I] = I, devuelva dicha posición o retorne el valor 0 cuando no exista ninguna. La complejidad ha de ser menor estricta que la lineal.

```
int Decide (int v[], int li, int ls) {

   if (li > ls) //
      return -1;
   else {
      int md = (li + ls)/2;
      if (v[md] == md) return md;
      else if (v[md] < md) return Decide (v,li,md-1);
           else return Decide (v,md+1,ls);
   }
}
```

****
// Problema nº 14: ¿v[i] = w[i]?, v creciente, w decreciente,
// al menos uno en sentido estricto
****

Sean X e Y dos vectores de enteros sin elementos repetidos de dimensión *N*. Se sabe que X está ordenado crecientemente y que Y lo está decrecientemente, al menos uno en sentido estricto. Construya una función recursiva que decida en tiempo logarítmico si hay un índice i tal que

X[i] = Y[i]

Construya una versión iterativa de dicha función.

```
int Decide (int v[], int w[], int li, int ls) {

   if (li > ls) // no existe
      return -1;
   else {
      int md = (li + ls)/2;
      if (v[md] == w[md]) return md;
      else if (v[md] < w[md]) return Decide (v,w,md+1,ls);
           else return Decide (v,w,li,md-1);
   }
}
```

****
// Problema nº 19
// Programa del rey MIDAS: Suponemos que la falsa pesa menos
****

El rey Midas se hizo famoso por su avaricia, que le llevó en cierta ocasión a proponerles a los sabios del reino el siguiente juego:

Tenemos una bolsa de N monedas de oro entre las que sabemos que hay una falsa, cuyo peso es menor que las demás. Teniendo presente que únicamente disponemos de una balanza de DOS PLATOS perfectamente equilibrada para pesarlas, diseñe una estrategia Divide y Vencerás segura, que encuentre la moneda falsa en el menor número de pesadas

NOTA: Se ha adoptar una estrategia del tipo divide y vencerás, aunque el número de pesadas se reduce por una leve observación. Estudie pues bien la subdivisión del problema.

¿Es realmente óptima la estrategia adoptada?

```
int Midas (int v[], int comienzo, int tam) {
   int tamSP, pesada;
   if (tam == 1) // hemos encontrado la falsa
      return comienzo;
   else {
      if (tam % 3 == 0) tamSP = tam/3;
      else tamSP = tam/3 + 1;
      pesada = Pesar (v,comienzo,comienzo+tamSP,tamSP);
      if (pesada == -1) return Midas (v,comienzo,tamSP);
      else if (pesada == 0)
               return Midas (v,comienzo+2*tamSP,tam-2*tamSP);
           else return Midas (v,comienzo+tamSP,tamSP);
   }
}


int Pesar (int v[], int com1, int com2, int tam) {
   int i, peso1 = 0, peso2 = 0;
   for (i=com1; i < com1+tam; i++) peso1 += v[i];
   for (i=com2; i < com2+tam; i++) peso2 += v[i];
   if (peso1 < peso2) return (-1);
   else if (peso1 == peso2) return 0;
   else return 1;
}
```

****

// Problema nº 3
// MEDIANA DE DOS VECTORES ORDENADOS

****

Sean A y B dos vectores ordenados de *n* elementos. Diseñe un algoritmo "Divide y Vencerás" de complejidad menor estricta que lineal que calcule la mediana del vector ordenado que resultaría de mezclar A y B.

```
int Mediana (int a[], int b[], int ca, int cb, int tam) {
   int ma, mb;
   if (tam == 1) return (Minimo(a[ca],b[cb]));
   else
      if (tam == 2)
         return (Minimo(Maximo(a[ca],b[cb]),Minimo(a[ca+1],b[cb+1]));
      else if (tam % 2 == 0) {
         tam = tam/2;
         ma = ca + tam - 1; mb = cb + tam;
         if (a[ma] == b[mb]) return (a[ma]);
         else if (a[ma] < b[mb]) return (Mediana(a,b,ma,cb,tam+1));
         else return (Mediana(a,b,ca,mb,tam));
      }
      else {
         tam = tam/2 +1;
         ma = ca + tam - 1; mb = cb + tam - 1;
         if (a[ma] == b[mb]) return (a[ma]);
         else if (a[ma] < b[mb]) return (Mediana(a,b,ma,cb,tam));
         else return (Mediana(a,b,ca,mb,tam));
      }
   }
```

```
*****************************************************************************
// Problema n° 16
// Selección del K-ésimo menor elemento de un vector
*****************************************************************************
```

Construya un procedimiento que seleccione el *K*-ésimo menor elemento de un vector de *N* elementos dado. La complejidad del procedimiento debe ser lineal.

```
void Selecciona (Type v[], int li, int ls, int k) {
// Suponemos que v[ls+1] >= v[k], li <= k <= ls
// Selecciona el k-ésimo menor elemento del vector v[1..n]
// y lo devuelve en la posición k, v[n+1] >= v[p], 1 <= p <= n
// si tenemos el vector v[o..n-1], será el de la posición k-1
// y se debe tener que v[n] >= v[p], 0 <= p <= n-1
   if (li == ls) { ;
      // No hay que hacer nada, está en la posición k = li = ls }
   else {
      int pos = Divide (v,li,ls+1);
      if (pos == k) { ;
         // No hay que hacer nada, está en la posición k=li=ls }
      else if (k > pos) Selecciona (v,pos+1,ls,k);
      else Selecciona (v,li,pos-1,k);
   }
}
```

Ampliación de la Programación

Divide y Vencerás

Girar Matriz

Girar Matriz

Enunciado

Dada una matriz cuadrada de tamaño 2^n, con n>0. Diseñe utilizando divide y vencerás, un algoritmo para rotarla 90º en el sentido de las agujas del reloj. Por ejemplo:

a	*b*	*c*	*d*
e	*f*	*g*	*h*
i	*j*	*k*	*l*
m	*n*	*ñ*	*o*

Se transforma en:

m	*i*	*e*	*a*
n	*j*	*f*	*b*
ñ	*k*	*g*	*c*
o	*l*	*h*	*d*

Girar Matriz

Enunciado

Dada una matriz cuadrada de tamaño 2^n, con $n > 0$. Diseñe utilizando divide y vencerás, un algoritmo para rotarla 90º en el sentido de las agujas del reloj. Por ejemplo:

a	b	c	d
e	f	g	h
i	j	k	l
m	n	ñ	o

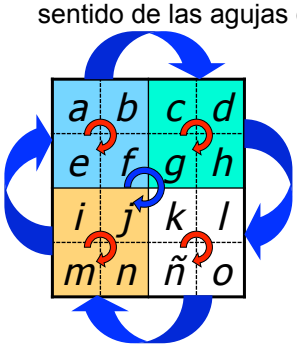
Se transforma en:

m	i	e	a
n	j	f	b
ñ	k	g	c
o	l	h	d

Girar Matriz

Enunciado

Dada una matriz cuadrada de tamaño 2^n, con $n > 0$. Diseñe utilizando divide y vencerás, un algoritmo para rotarla 90º en el sentido de las agujas del reloj. Por ejemplo:



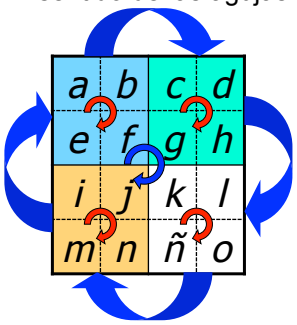
Se transforma en:

m	i	e	a
n	j	f	b
ñ	k	g	c
o	l	h	d

Girar Matriz

Enunciado

Dada una matriz cuadrada de tamaño 2^n, con $n > 0$. Diseñe utilizando divide y vencerás, un algoritmo para rotarla 90º en el sentido de las agujas del reloj. Por ejemplo:

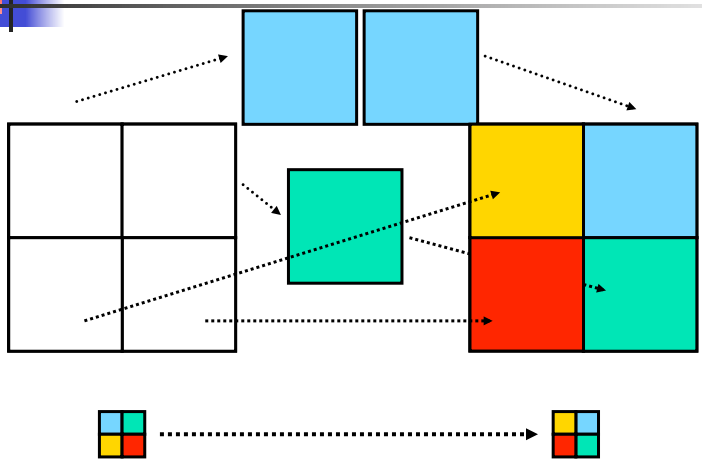


Otra forma:

Se transforma en:

m	i	e	a
n	j	f	b
ñ	k	g	c
o	l	h	d

Girar Matriz



Girar Matriz

Resolución

```
void Girar (int Origen [MAXDIM][MAXDIM], int Girada [MAXDIM][MAXDIM],
            int Fila, int Columna, int FilaGirada, int ColGirada, int N) {

// Giramos la submatriz de la matriz Origen que comienza en [Fila,Columna]
// y la ponemos en la submatriz de Traspuesta en la posición [FilaTrasp,ColTrasp],
// ambas de dimensión N

  if (N == 1) // Caso Base
    Girada[FilaGirada][ColGirada] = Origen[Fila][Columna];
  else {
    Girar (Origen, Girada, Fila, Columna, FilaGirada, ColGirada+N/2, N/2);
    Girar (Origen, Girada, Fila+N/2, Columna, FilaGirada, ColGirada, N/2);
    Girar (Origen, Girada, Fila+N/2, Columna+N/2, FilaGirada+N/2, ColGirada, N/2);
    Girar (Origen, Girada, Fila, Columna+N/2, FilaGirada+N/2, ColGirada+N/2, N/2);
  }

}
```

Ampliación de la Programación

Divide y Vencerás

Trasponer

Traspuesta

Enunciado

Dada una matriz cuadrada de tamaño 2^n, con $n > 0$. Diseñe utilizando ***divide y vencerás***, un algoritmo para calcular su traspuesta. Por ejemplo:

a	*b*	*c*	*d*
e	*f*	*g*	*h*
i	*j*	*k*	*l*
m	*n*	*ñ*	*o*

Se transforma en:

a	*e*	*i*	*m*
b	*f*	*j*	*n*
c	*g*	*k*	*ñ*
d	*h*	*l*	*o*

Traspuesta

Enunciado

Dada una matriz cuadrada de tamaño 2^n, con $n > 0$. Diseñe utilizando ***divide y vencerás***, un algoritmo para calcular su traspuesta. Por ejemplo:

a	b	c	d
e	f	g	h
i	j	k	l
m	n	ñ	o

Se transforma en:

a	e	i	m
b	f	j	n
c	g	k	ñ
d	h	l	o

Traspuesta

Enunciado

Dada una matriz cuadrada de tamaño 2^n, con $n > 0$. Diseñe utilizando ***divide y vencerás***, un algoritmo para calcular su traspuesta. Por ejemplo:

a	b	c	d
e	f	g	h
i	j	k	l
m	n	ñ	o

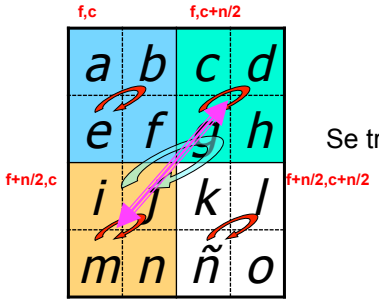
Se transforma en:

a	e	i	m
b	f	j	n
c	g	k	ñ
d	h	l	o

Traspuesta

Enunciado

Dada una matriz cuadrada de tamaño 2^n, con $n > 0$. Diseñe utilizando ***divide y vencerás***, un algoritmo para calcular su traspuesta. Por ejemplo:



Se transforma en:

a	e	i	m
b	f	j	n
c	g	k	ñ
d	h	l	o

Traspuesta

Resolución

```
void Trasponer (int Origen [MAXDIM][MAXDIM], int Traspuesta [MAXDIM][MAXDIM],
                int Fila, int Columna, int FilaTrasp, int ColTrasp, int N) {

// Trasponemos la submatriz de la matriz Origen que comienza en [Fila,Columna]
// y la ponemos en la submatriz de Traspuesta en la posición [FilaTrasp,ColTrasp],
// ambas de dimensión N

  if (N == 1) // Caso Base
    Traspuesta[FilaTrasp][ColTrasp] = Origen[Fila][Columna];
  else {
    Trasponer (Origen, Traspuesta, Fila, Columna, FilaTrasp, ColTrasp, N/2);
    Trasponer (Origen, Traspuesta, Fila+N/2, Columna, FilaTrasp, ColTrasp+N/2, N/2);
    Trasponer (Origen, Traspuesta, Fila, Columna+N/2, FilaTrasp+N/2, ColTrasp, N/2);
    Trasponer (Origen, Traspuesta, Fila+N/2, Columna+N/2, FilaTrasp+N/2, ColTrasp+N/2, N/2);
  }

}
```

```
Algoritmo Tornillos(tornillos,tuercas:vector, inicio,fin:enteros)
Variable
        pTornillos, pZazaptos: entero;

Inicio
        Si inicio <= fin entonces
                pTornillos = tomarPivoteTornillos(tornillos,inicio,fin);
                partir(tuercas,tornillos[pTornillos],inicio,fin);
                pTuercas = tomarPivoteTuercas(tuercas,tornillos[pTornillos],inicio,fin);
                si (pTuercas <> -1) partir(tornillos,tuercas[pTuercas],inicio,fin);
                Tornillos(tornillos,tuercas,inicio,pTornillos);
                Tornillos(tornillos,tuercas,pTornillos+1,fin);
        Fin_entonces
Fin

Int tomarPivoteTornillos(tornillos:vector, inicio,fin:entero): entero
Inicio
        Devolver inicio;
Fin

Int tomarPivoteTuercas(tuercas:vector, pTornillos,inicio,fin:entero):entero
Variables
        Pos, valor: entero
Inicio
        Pos = inicio;
        Mientras ((pos<=fin) && (tuercas[pos]<>pTornillos))
                Pos++
        Fin_mientras

        Si (pos>fin) valor = -1;
        Sino valor =pos;
        Devolver pos;
fin
```