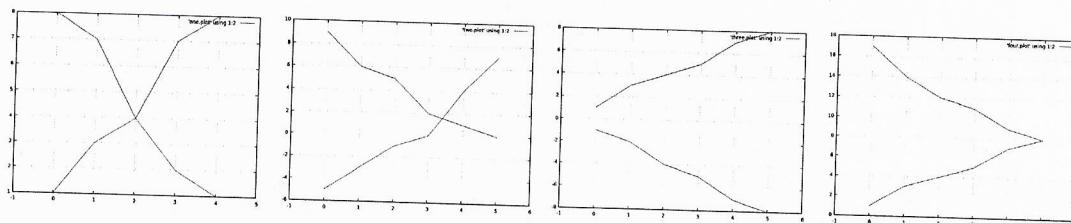


Soluciones
REC ENE 2019

Problema 2

Problema 2 (3 puntos) Tenemos dos vectores de números enteros, de la misma longitud ($N > 1$), ordenados, el primero en orden estrictamente creciente y el segundo en orden estrictamente decreciente. Suponiendo que representan sendas líneas poligonales encuentra el intervalo $[n, m]$ en el que se encuentra el punto de cruce. Por convenio, si el punto *virtual* en que se cruzan es anterior o posterior al comienzo o final de las líneas se devolverán los intervalos $[-1, 0)$ y $[N - 1, N)$ respectivamente como intervalos de cruce.

1. Implementa un algoritmo que resuelva el problema aplicando la técnica de divide y vencerás. El coste debe ser lo más eficiente posible.
2. Indicar el coste de la solución obtenida y justificarlo.



Entrada

La entrada consta de una serie de casos de prueba. Cada caso de prueba consta de tres líneas, en la primera se muestra un valor que representa el número de elementos de los dos vectores. En la línea siguiente se muestran los valores del primer vector y en la tercera línea los valores del segundo vector. La entrada termina con una línea con un cero.

Salida

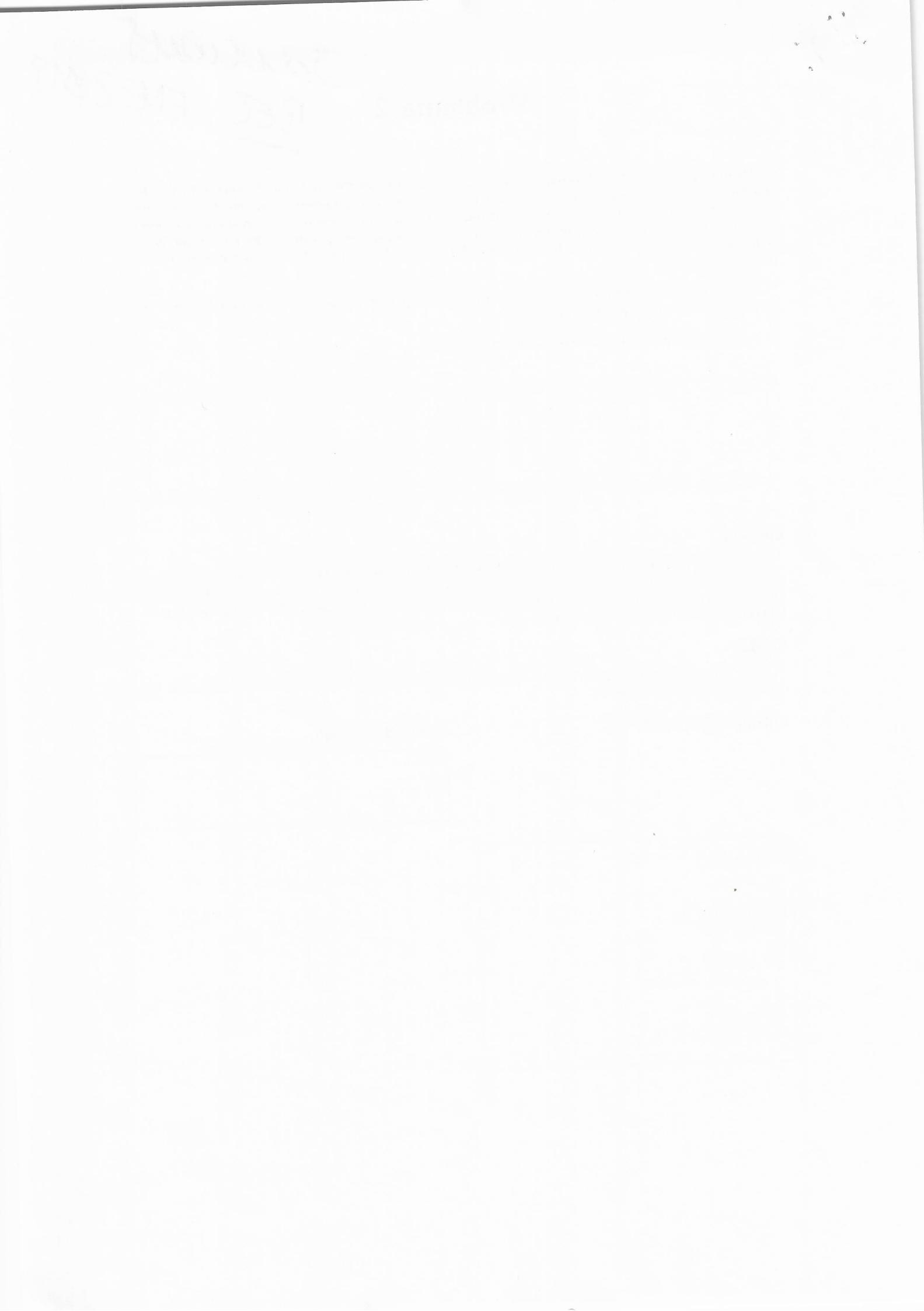
Para cada caso de prueba se escribe en una línea el intervalo en el que coincide el punto con los convenios señalados más arriba.

Entrada de ejemplo

```
5
1 3 4 7 8
8 7 4 2 1
6
-5 -3 -1 0 4 7
9 6 5 2 1 0
6
1 3 4 5 7 8
-1 -2 -4 -5 -7 -8
6
1 3 4 5 7 8
17 14 12 11 9 8
0
```

Salida de ejemplo

```
2 3
3 4
-1 0
5 6
```



/*

Intersection positions on strict polygonal lines.

(Spec. Not required)

```
P : N-0>=2 strictInc(V[0..N]) and strictDec(W[0..N])
cross(V[0..N],W[0..N]) return n
Q : n = max p : 0 <= p + 1 <= N and (p >= 0 -> V[p]<=W[p]):p
** N-0 is on purpose for syntatic replacement.
```

Visual Snapshot: (hard to depict in ASCII)

Our point lies between on V[i..j) or (-i-1,i) or (j-1,j)

Tail approach Immersion:

```
P' : Q[N-0/j-i,0/i,N/j] and 0<=i < j <= N
crossG(V[i..j],W[i..j]) return n
Q' : Q
```

Init call

crossG(V[0..N],W[0..N])

Pseudocode:

```
let h = (i+j)/2
    V[h]=W[h] -> h (Basis)
    V[h]<W[h] ->
        (j - h) >= 2 -> crossG(V,W,h,j) (Recursive)
        e.o.c -> h (Basis)
    V[h]>W[h] ->
        (h-i) >= 2 -> crossG(V,W,i,h) (Recursive)
        e.o.c ->
            V[i]>W[i] -> i - i (Basis)
            V[i]<=W[i] -> i (Basis)
```

Proof: (Not required)

Note: numberes lines must computer-aided proof.

- Sound immersion:

1 .- P' and i=0 and j=N and Q' |- 0

- Full cases range

2 .- P' |- (V[h]=W[h] ||
V[h]<W[h] and (j-h>=2 || j-h < 2) ||
V[h]>W[h] (h-i>=2 || h-i < 2))

- Success for Trivial

3 .- P' and V[h]=W[h] |- Q'

4 .- P' and V[h]<W[h] and j-h<2 |- Q'

5 .- P' and V[h]>W[h] and h-i<2 and V[i]>W[i] |- Q'

6 .- P' and V[h]>W[h] and h-i<2 and V[i]<=W[i] |- Q'

- Success for recursive:

7 .- P' and V[h]<W[h] and (j-h>=2) and Q' |- Q'

8 .- P' and V[h]>W[h] and (h-i>=2) and Q' |- Q'

- Positive defined Quota:
9.- $P' \mid (j-i) \geq 0$
- Decreasing quota:
10 .- P' and $V[h] < W[h]$ and $(j-h \geq 2) \mid (j-h) < (j-i)$
11 .- P' and $V[h] > W[h]$ and $(h-i \geq 2) \mid (h-i) < (j-i)$
- Init call:
9 .- $P'[i/0, j/N]$

According to Theorem of Division, A=1, B=2, and k=0, hence $O(\log(n))$

```
/*
#include <iostream>
#include <algorithm>

using namespace std;

#define MAX 1000

/* Immersion */
int crossG(const int V[], const int W[], const int N,
           const int i, const int j)
{
    // cout << i << " " << j << endl;
    const int h = (i+j)/2;
    if (V[h]==W[h]) return h;
    if (V[h]<W[h])
    {
        if ((j-h)>=2) return crossG(V,W,N,h,j);
        return h;
    }
    if (V[h]>W[h])
    {
        if ((h-i)>=2) return crossG(V,W,N,i,h);
        if (V[i]>W[i]) return i-1;
        return i;
    }
    return -5; // (Never reached)
}

void cross(const int V[], const int W[], const int N, int &n, int &m)
{
    n = crossG(V,W,N,0,N);
    m = n + 1;
    return ;
}

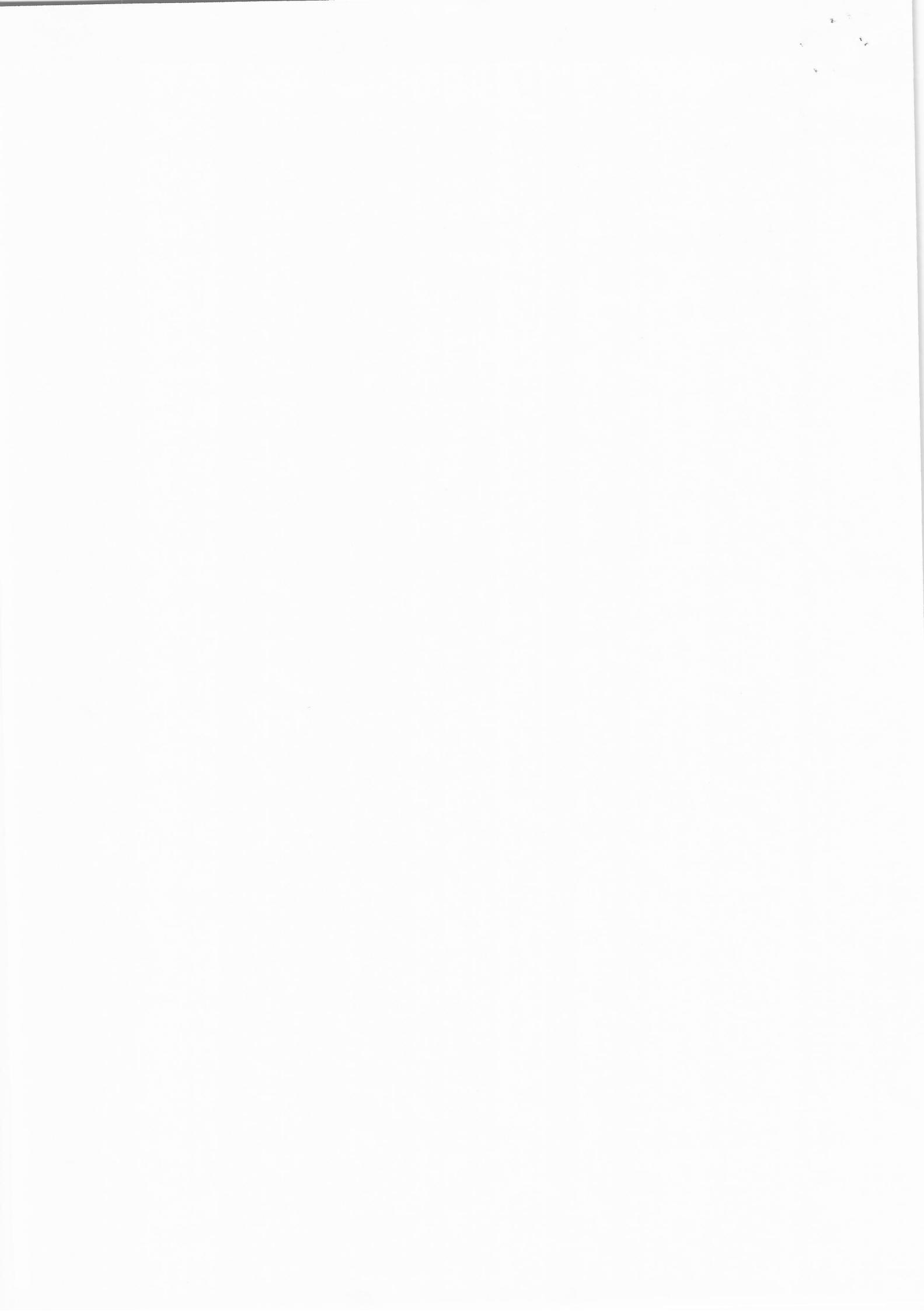
int main(int argc, char *args[])
{
    int N;
    int V[MAX], W[MAX];
    int n, m;
    for (cin >> N ; N; cin >> N)
    {
        for (int n=0; n<N; n++) cin >> V[n];
        for (int n=0; n<N; n++) cin >> W[n];
        cross(V,W,N,n,m);
        cout << n << " " << m << endl;
    }
    return 0;
}

/*
```

5
1 3 4 7 8
8 7 4 2 1
2 3
6
-5 -3 -1 0 4 7
9 6 5 2 1 0

3 4
6
1 3 4 5 7 8
-1 -2 -4 -5 -7 -8
-1 0
6
1 3 4 5 7 8
17 14 12 11 9 8
5 6
0

*/



- 2. (3 puntos)** Se dice que un entero positivo es “sumdivisible” si la suma de sus dígitos es divisible por el número de dígitos y al quitar el último, el resultado es también “sumdivisible”. Por ejemplo, el número 33374 es “sumdivisible”, ya que:

- 33374 tiene 5 dígitos y $3+3+3+7+4=20$ es divisible por 5
- 3337 tiene 4 dígitos y $3+3+3+7=16$ es divisible por 4
- 333 tiene 3 dígitos y $3+3+3=9$ es divisible por 3
- 33 tiene 2 dígitos y $3+3=6$ es divisible por 2
- 3 tiene 1 dígito y $3=3$ es divisible por 1

Se pide:

- (a) **(2,5 puntos)** Implementar un algoritmo recursivo que tome como entrada un entero positivo y determine si es o no “sumdivisible”
 (b) **(0,5 puntos)** Determinar justificadamente el orden de complejidad del algoritmo

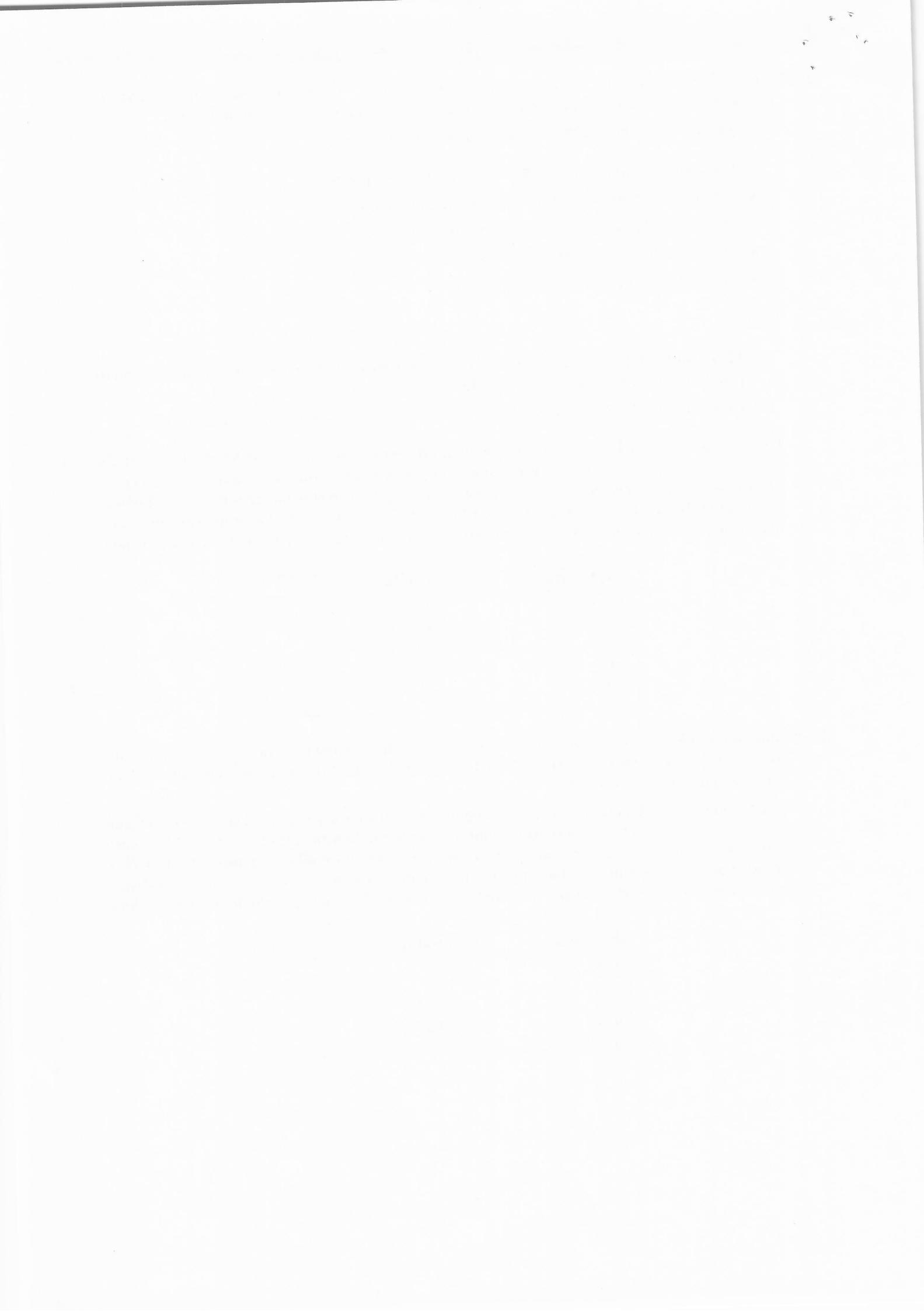
La implementación deberá ir acompañada de un programa de prueba, que lea desde la entrada estandar casos de prueba, los ejecute, e imprima por la salida estandar el resultado. Cada caso de prueba será una linea con un entero positivo. La salida correspondiente será SI si el número es “sumdivisible”, y NO en caso contrario. El final de los casos de prueba se indicará mediante una linea que contiene únicamente 0. A continuación se muestra un ejemplo de entrada / salida:

Entrada	Salida
33374	SI
33373	NO
84	SI
85	NO
9	SI
0	

- 3. (3 puntos)** Implementar un algoritmo de “vuelta atrás” que, tomando como entrada (i) un dígito positivo D; (ii) un entero positivo K, devuelva la cantidad total de números “sumdivisibles” de K dígitos que comienzan por D.

La implementación deberá ir acompañada de un programa de prueba, que lea desde la entrada estandar casos de prueba, los ejecute, e imprima por la salida estandar el resultado. Cada caso de prueba será una linea con los valores de D y K en este orden. La salida correspondiente será la cantidad de números “sumdivisibles” pedida. El final de los casos de prueba se indicará mediante una linea que contiene únicamente 0. A continuación se muestra un ejemplo de entrada / salida:

Entrada	Salida
1 2	5
2 3	16
5 8	112
9 20	150
0	



```

/*
Recursive : sumdivisible

As a decision problem, we model it as optimization problem. (min,max)
(Spec not required )

P : N > 0
sumDivisible(N) return b
Q : b = \max i : 0+0 <= i <= ceillog(N) and ((i>0)-> (sum(N,i,ceillog(N)) % i) != 0):i *
* 0+0 is on purpose to make syntactic expression substitution.

```

where $\text{ceillog}(N) = 1 + \min i : 0 \leq i : 10^i \leq N$
 (Informally, ceilog marks the number of cyphres)

$\text{sum}(N,i,A) = \sum j : A-i \leq j < A : (N/10^{(\log(N)-j)}) \% 10$
 (Informally Summing up to i digits)

Visual snapshot

 33374 -> 1 (YES)

```

3 + 3 + 3 + 7 + 4 = 20 = sum(33374,5,5) \% 5 =0
3 + 3 + 3 + 7      = 16 = sum(33374,4,5) \% 5 =0
3 + 3 + 3          = 9  = sum(33374,3,5) \% 5 =0
3 + 3              = 6  = sum(33374,2,5) \% 5 =0
3                  = 3  = sum(33374,1,5) \% 5 =0

```

33373 -> 0 (NO)

3 + 3 + 3 + 7 + 4 = 19 = sum(33374,5,5) \% 5 !=0

Tail approach Immersion:

```

P' : 0<= n <= ceillog(N) and
0 <= NN <= N and NN=N/10^(ceillog(N)-n)
ac = sum(N,n,ceillog(N)) ** efficiency reasons.
n<ceillog(N) -> Q[0+0/n+1]
sumDivisibleG(N,NN,n,ac) return b
Q' : Q

```

** efficiency reasons.

Init call

```

-----  

let
  ac = sum(N,0,log(N))
  n  = ceillog(N)
in
sumDivisibleG(N,N,n,ac)

```

Pseudocode:

General case:

$n > 0$ and $ac \% n == 0$ -> $\text{sumDivisibleG}(N, NN/10, n-1, ac - (NN \% 10))$

Basis case:

$n == 0$ -> n
 $n > 0$ and $ac \% n != 0$ -> n

According to Theorem of Division, $A=1$, $B=2$, and $k=0$, hence $O(\log(n))$

Proof: (Not required)

Note: numberes lines must computer-aided proof.

Note: warning typos

- Sound immersion:

1 .- P' and n=ceillog(N)) and and N>NN and ac=sum(N,n,n) and Q' |- Q

- Full cases range

2 .- P' |- (n==0 || ((n>0) && (ac % n)!= 0 || (ac % n)== 0))

- Success for Trivial:

3 .- P' and n==0 |- Q'

4 .- P' and n > 0 and (ac % n)!=0 |- Q'

- Precondition holds after recursive call.

5 .- P' and n > 0 and (ac % n)!=0 |- P'[NN/(NN/10),n/n-1,ac/ac-(NN%10)]

- Success for Recursive:

6 .- P' and n > 0 and (ac % n)!=0 and Q' |- Q'

- Positive defined Quota :

7 .- P' |- NN >= 0

- Decreasing quota

8 .- P' and n > 0 and (ac % n)!=0 |- NN/10 <= N

- Init call:

9 .- |- P'[NN/N,n/ceillog(n),ac=sum(N,ceillog(n),ceillog(n))]

*/

```
#include <iostream>
#include <algorithm>

using namespace std;

/* first parameter can be omitted. Only for predicates*/
int sumDivisibleG(const int N, const int n, const int ac)
{
    if (!n) return n;
    if (n && (ac%n)) return n;
    // (General case : n>0 and ac%n==0)
    return sumDivisibleG(N/10,n-1,ac-(N%10));
}

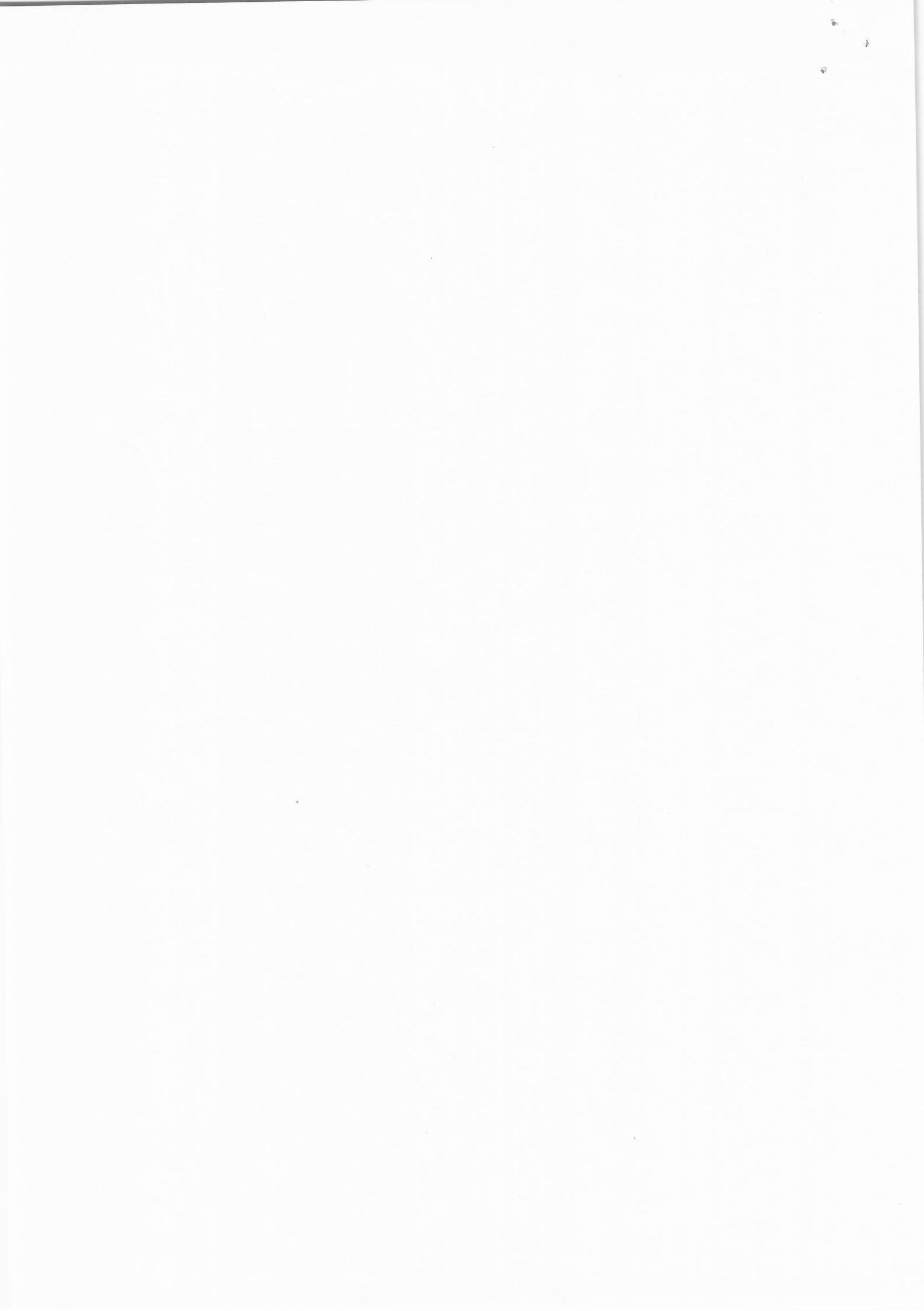
int sumDivisible(const int N)
{
    int n,ac;
    int NN = N;
    for( ac=NN%10, n=1 ; NN/10 ; NN/=10) { n++; ac+=(NN/10)%10 ; } // log(N)
    return (0==sumDivisibleG(N,n,ac)); // log(N)
}

int main(int argc, char *args[])
{
    int N;
    for(cin >> N; N ; cin >> N)
        cout << sumDivisible(N) << endl;
}

// cout << " N " << N << " n " << n << " ac " << ac << endl;
/*
```

```
> g++ 02Recursion.rafa.cpp -o main && ./main
33374
1
33373
0
84
1
85
0
9
1
0
*/

```



2.(2.5 puntos) Dado un vector formado por una secuencia de 1 seguido de una secuencia de 0, se desea averiguar el número de 0s que contiene. Se pide:

1. (1.5 puntos) Escribe un algoritmo recursivo eficiente (mejor que lineal) que permita resolver el problema para un vector dado.
2. (1 punto) Escribe la recurrencia que corresponde al coste de la función recursiva e indica a qué orden de complejidad asintótica pertenece dicho coste.

Entrada

La entrada comienza con una línea que contiene el número de casos de prueba. Cada caso de prueba contendrá en una línea el tamaño del vector y a continuación en la siguiente línea los valores que contiene el vector.

Recuerda: el número de 0s no se ha de calcular durante la lectura de los datos.

Salida

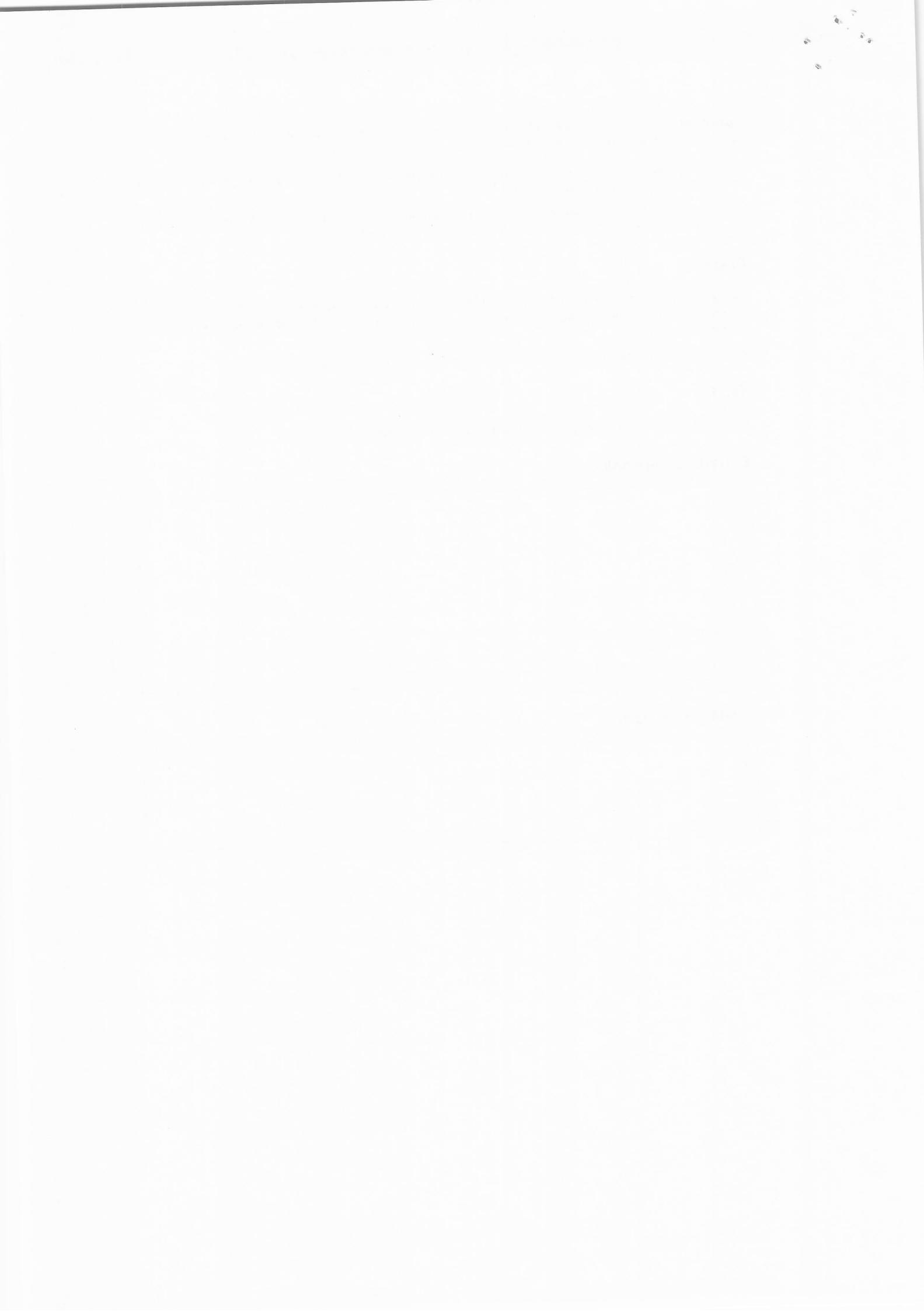
Por cada caso de prueba el programa escribirá el número de 0s.

Entrada de ejemplo

```
6
0
1
1
2
1 0
2
1 1
10
1 1 1 1 1 1 0 0 0
10
0 0 0 0 0 0 0 0 0
```

Salida de ejemplo

```
0
0
1
0
3
10
```



```

/*
Recursive: Find number of 0 is
a decreasing sequence of 1 and 0.s
in log(N)

0
+-----+-----+-----+-----+-----+-----+
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
+-----+-----+-----+-----+-----+-----+
n = 2
*/

```

/*

```

P : decr(V,0,N)
zeros(V[0..N of {0,1}) returns n
Q : n = #i : 0 <= i < N : V[i]==0

```

tail approach Immersion:

```

P': dec(V,0,N) and
0 <= i <= j <= N and
ac = #p : 0 <= p < i : V[p]==0 +
#q : j <= q < N : V[q]==0
zerosG(V[i..j],N,ac) returns n
Q': Q

```

P' Visual snapshot:

```

ac = 4
i=0
          j
+-----+-----+-----+-----+-----+-----+
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
+-----+-----+-----+-----+-----+-----+

```

Init call:

```
zerosG(V[0..N],N,0);
```

Pseudocode:

General case:

$(j-i) \geq 2$

```

V[h]==0      : zeros(V,i,h,ac+(j-h))
V[h]==1      : zeros(V,h,j,ac)

```

Basis cases:

```

(j-i)==0 : return ac
(j-i)==1 : return ac + V[i]==1

```

Proof: (not required)

Note: numbered lines must be computer-aided proof.

- Sound immersion:

1 .- P' and i=0 and j=N and Q' |- Q

- Full cases range

- 2 .- $P' \mid (j-i) \geq 2 \text{ or } (j-i) < 2$

- Success for Trivial:

- 3 .- P' and $i-j=0 \mid Q'$
- 4 .- P' and $i-j=1 \mid Q'$

when $0 \leq i < j+1 \leq N$, let $h = (i+j)/2$. Then

- Precondition holds after recursive call.

- 5 .- P' and $i-j \geq 2$ and $V[h] == 0 \mid P'[j/h, ac/ac+(j-h)]$
- 6 .- P' and $i-j \geq 2$ and $V[h] == 1 \mid P'[i/h]$

- Success for Recursive:

- 7 .- P' and $i-j \geq 2$ and $V[h] == 0$ and $Q' \mid Q'$
- 8 .- P' and $i-j \geq 2$ and $V[h] == 1$ and $Q' \mid Q'$

- Positive defined Quota :

- 9 .- $P' \mid -(j-i) \geq 0$

- Decreasing quota

- 10 .- P' and $i-j \geq 2$ and $V[h] == 0 \mid (h-i) < (j-i)$
- 11 .- P' and $i-j \geq 2$ and $V[h] == 1 \mid (j-h) < (j-i)$

- Init call:

- 12 .- $\mid P'[0/i, N/j, ac/0]$

*/

```
/* Complexity, log(N), A=1 B=2 and k=0 */
int zerosG(const int V[], const int N,
           const int i, const int j,
           const int ac)
{
    if (i==j) return ac;
    if (j==i+1) return ac + (V[i]==0);
    const int h = (i+j)/2;
    if (V[h]==0) return zerosG(V,N,i,h,ac+(j-h));
    else return zerosG(V,N,h,j,ac);
}

int zeros(const int V[], const int N)
{
    return zerosG(V,N,0,N,0);
}

#include <iostream>

using namespace std;

#define MAX 10000
int main(int argc, char *args[])
{
    int C,c,N,n;
    int V[MAX];
    cin >> C;
    for(c=0; c<C; c++)
    {
        cin >> N ;
        for(n=0; n<N; n++) cin >> V[n];
        cout << zeros(V,N) << endl;
    }
}
```