

## SOPA DE LETRAS

Archivo de entrada `sopaent.txt`

Archivo de salida `sopasal.txt`

Sopa de letras es un juego, donde se tiene que buscar en una retícula llena de letras, una lista de palabras.

### Problema

Dada una retícula con letras, y una lista de palabras, debes encontrar el lugar donde se encuentra cada palabra así como su dirección, o en su caso decir que dicha palabra no se encuentra.

### Entrada

El primer renglón tendrá dos enteros positivos ( $R$ ,  $C$ ) no mayores a 100, serán la cantidad renglones y columnas de la retícula respectivamente. Después seguirán  $R$  renglones con  $C$  caracteres cada uno, y por último la palabra a buscar.

Todas las letras y palabras estarán en mayúsculas. Cada palabra estará en la retícula a lo más una vez. Las palabras no tendrán más de veinte letras.

### Salida

Un renglón con el siguiente formato:

```
palabra r c orientación
```

Donde cadena es la palabra leída,  $r$  y  $c$  serán el renglón y columna en la que se encuentra la primera letra de la palabra, y orientación será: N, NE, E, SE, S, SO, O, NO, dependiendo si la palabra está orientada al Norte, Noreste, Este, Sureste, Sur, Suroeste, Oeste o Noroeste respectivamente. En caso de que la palabra no se encuentre el formato deberá ser: `palabra NO SE ENCUENTRA`

### Ejemplo de entrada

```
6 5  
KLRTY  
PAGSD  
QURJR  
GEEEE  
GBUJL  
HOLAK  
KAREL
```

### Ejemplo de salida

```
KAREL 1 1 SE
```

**Ejemplo de entrada**

6 5  
KLRTY  
PAGSD  
QURJR  
GEEEE  
GBUJL  
HOLAK  
CHI

**Ejemplo de salida**

CHIH NO SE ENCUENTRA

**Secretos**

Las técnicas de encriptación son muy variadas, un técnico ha propuesto la siguiente: Dado un texto y una clave, la clave se sobrepone al inicio del texto ocasionando que las letras que forman al texto se sustituyan por otras ubicadas tantos caracteres a la derecha como la posición de la letra que conforma la clave en el alfabeto; luego se suman las posiciones en el alfabeto de las letras modificadas y tomamos el último dígito de la suma, este se utiliza para aumentar el desplazamiento de la siguiente pasada.

**Ejemplo**

clave: **abcd**

texto: **nomevisiteshoy.**

(1)

abcd

**nomevisiteshoy**

**oqpi**

suma (o+q+p+i)= 57

(2)

\*\*\*\*abcd

nome**visi**teshoy

\*\*\*\***drct**

suma (d+r+c+t)= 45

(3)

\*\*\*\*\*abcd

nomevisit**eshoy**

\*\*\*\*\***zlaq**

suma (z+l+a+q)= 56

(4)

\*\*\*\*\*abcd

nomevisiteshoy

\*\*\*\*\***vg**

El mensaje encriptado es:

**oqpidrctzlaqvq**

**Problema.**

Dado un mensaje y una clave, encriptar el mensaje.

**Entrada.**

Un archivo con dos renglones, en el primero viene la clave y en el segundo el mensaje. La longitud máxima del mensaje es de 200 letras, solo contiene letras minúsculas. La longitud máxima de la clave será de 20 letras.

**Salida.**

Un archivo con el mensaje encriptado.

**Ejemplo**

secreto.in

**omi  
isaackarelov**

secreto.out

**xfjpptrgpbcf**

## Vaquitas

Cierto día, Pepe el Toro (amo de las tierras vacunas) ordenó que todas las vacas de su reino formaran una larga fila para empadronarlas. Así pues, todas las vacas se formaron en fila para agradar a su rey. Hecho lo anterior, le asignó un número entero a cada vaca, después le dijo a su más fiel sirviente: “Si eres capaz de calcular el valor máximo que se obtiene al sumar los números que le puse a cada Vaca y que están en forma consecutiva, te regalaré un viaje todo pagado a la 11ª OMI.” Ayuda al lacayo a encontrar dicho valor.

Problema:

Dada una lista de números enteros, encontrar la suma máxima de un conjunto de números sucesivos.

Entrada:

La primera línea tendrá un entero positivo  $n$ ,  $1 \leq n \leq 30000$ , la cantidad de vacas. La siguiente línea tendrá los valores asignados a cada Vaca, separados por espacios. Cada entero estará entre  $-30000$  y  $30000$ .

Salida:

Un entero, la suma máxima de los números consecutivos. Este entero no sobrepasará a  $1000000000$ .

Ejemplo de Entrada:

```
9  
3 2 -7 3 -6 5 6 4 -8
```

Ejemplo de Salida

```
15
```

Archivo de entrada vaca.ent  
Archivo de salida vaca.sal

## La permutación

Una permutación de un conjunto de elementos, donde los elementos son:  $a_1, a_2, \dots, a_i, \dots, a_j, \dots, a_n$ , puede ser  $a_1, a_2, \dots, a_j, \dots, a_i, \dots, a_n$ . Es decir una permutación de un conjunto de elementos es un conjunto donde se tienen los mismos elementos en diferente orden.

Por otro lado sabes que una permutación  $a$  es lexicográficamente menor que la permutación  $b$  cuando se tiene que:

Para todo elemento en las posiciones anteriores a  $i$  los elementos en cada posición son iguales y en la posición  $i$  el elemento en  $a$  es menor al elemento en  $b$ .

Es decir:

$$a_1 = b_1, a_2 = b_2, \dots, a_i < b_i, \dots$$

Por ejemplo "123" es menor que "132", La permutación "aaab" es menor que la permutación "abaa".

### Problema

Dados dos números enteros,  $n$  y  $k$  donde  $n$  es el número de elementos del conjunto, donde  $1 < n < 13$ , y los elementos del conjunto serán los número enteros del 1 al  $n$ , deberás obtener la  $k$ -ésima permutación en orden lexicográfico.

### Entrada

Deberás de leer de archivo "permuta.in" dos números enteros  $n$  y  $k$ , donde  $1 < n < 13$ , y  $0 < k < n!$

### Salida

Debes imprimir la  $k$ -ésima permutación de los  $n$  elementos separados por espacios en el archivo "permuta.sal".

*Nota :  $n!$  ( $n$  factorial) tiene la siguiente definición:*

$$n! = \begin{cases} 1 & \text{si } n = 1 \text{ ó } n = 0 \\ n (n-1)! & \text{En cualquier otro caso} \end{cases}$$

### Ejemplo 1

ENTRADA	SALIDA
4 17	3 4 2 1

### Ejemplo 2

ENTRADA	SALIDA
3 4	2 3 1

**Nota: El código de tu programa deberá llamarse, permuta.c, permuta.cpp ó permuta.pas, según sea el caso. Y tu ejecutable permuta.exe**

*Tu programa deberá correr en menos de 0.5 s*