
Codificar en Base64 (2)**W53973_es**

(Este problema utiliza la función `base64_to_char` del problema "Codificar en Base64 (1)".)

Se trata de hacer un programa que, dada una secuencia de bytes en la entrada (naturales entre 0 y 255), la **codifique en base 64**. La codificación funciona de la siguiente manera, y trabaja con tripletas de 3 bytes:

1. Primero, cada 3 bytes de entrada B_1 , B_2 y B_3 , construimos un entero x en base 256 de la siguiente manera:

$$x = (B_1 \cdot 256 + B_2) \cdot 256 + B_3$$

2. Despues, reinterpretamos x en base 64 y extraemos las cifras, que ahora son 4: d_1 , d_2 , d_3 , y d_4 . El proceso es totalmente análogo a extraer las cifras de un número en base 10, pero en base 64. Lo que estamos haciendo es calcular los dígitos d_i de la fórmula siguiente:

$$x = ((d_1 \cdot 64 + d_2) \cdot 64 + d_3) \cdot 64 + d_4$$

(Cabe recordar que el proceso de extracción de cifras trabajado en PRO1 produce las cifras al revés, es decir, comenzando por d_4 .)

3. Por último, usando la función `base64_to_char` convertimos d_1 , d_2 , d_3 y d_4 en caracteres y los mostramos en la salida en este orden.

Si la secuencia de entrada no tiene una longitud múltiple de 3, justo al final, tendremos un grupo de solo 1 o 2 bytes:

- Si tenemos un grupo de 2 bytes: asignamos $B_3 = 0$, y seguimos igualmente los pasos de la codificación. Una vez con las cifras d_i , cambiamos d_4 por el carácter '='.
- Si tenemos un grupo de 1 byte: asignamos $B_2 = 0$ y $B_3 = 0$, y seguimos igualmente los pasos de la codificación. Una vez con las cifras d_i , cambiamos tanto d_3 como d_4 por el carácter '='.

Entrada

La entrada consiste en varias secuencias de bytes. Cada una comienza con un entero n que indica el número de bytes que siguen y después hay n bytes, donde cada uno es solo un número natural entre 0 y 255 (ambos incluidos).

Salida

La salida debe ser una línea para cada caso con la codificación en base 64 de la secuencia de bytes de la entrada, sin espacios entre los caracteres.

Observaciones

- Este problema tiene como centro de interés la **corrección** y la **legibilidad**. Sobre la legibilidad, se valorará que el programa utilice funciones para evitar repetición y separar las diversas tareas.
- Si tenéis la función `base64_to_char` y el Juez os la ha aceptado, usadla directamente. Si no, copiad la siguiente definición:

```
char base64_to_char(int d) {
    static char _syms[65] =
        "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
        "abcdefghijklmnopqrstuvwxyz"
        "0123456789+/";
    return _syms[d];
}
```

Ejemplo de entrada 1

```
3 0 0 0
3 1 1 1
3 2 2 2
3 255 255 255
1 0
2 0 0
1 1
2 1 1
4 100 100 100 100
5 10 0 20 0 30
6 255 255 255 0 0 0
```

Ejemplo de salida 1

```
AAAA
AQEB
AgIC
////
AA==
AAA=
AQ==_
AQE=
ZGRkZA==
CgAUAB4=
////AAAA
```

Ejemplo de entrada 2

```
1 44
1 38
1 150
1 74
1 221
1 50
1 71
1 54
1 5
1 76
1 162
```

Ejemplo de salida 2

```
LA==
Jg==
lg==
Sg==
3Q==
Mg==
Rw==
Ng==
BQ==
TA==
og==
```

Ejemplo de entrada 3

```
2 246 161
2 198 147
2 111 211
2 13 73
2 8 125
2 172 240
2 145 41
2 193 95
2 109 8
2 102 246
```

Ejemplo de salida 3

```
9qE=
xpM=
b9M=
DUK=
CH0=
rPA=
kSk=
wV8=
bQg=
ZvY=
```

Ejemplo de entrada 4

```
3 187 86 98
3 171 169 241
3 52 145 94
3 182 60 181
3 212 26 182
3 253 228 97
3 171 64 63
3 216 173 75
3 0 213 97
3 122 15 167
```

Ejemplo de salida 4

```
u1Zi
q6nx
NJFe
t jy1
1Bq2
/eRh
q0A/
2K1L
ANVh
eg+n
```

Información del problema

Autor : Pau Fernández

Generación : 2025-10-30 12:08:33

© *Jutge.org*, 2006–2025.

<https://jutge.org>