
Codificar en Base64 (2)**W53973_ca**

(Aquest problema fa servir la funció `base64_to_char` del problema "Codificar en Base64 (1)".)

Es tracta de fer un programa que, donada una seqüència de bytes a l'entrada (naturals entre 0 i 255), la **codifiqui en base 64**. La codificació funciona de la següent manera, i treballa amb tripletes de 3 bytes:

1. Primer, cada 3 bytes d'entrada B_1 , B_2 i B_3 , construïm un enter x en base 256 de la següent manera:

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

2. Després, reinterpretem x en base 64 i n'extraïem les xifres, que ara són 4: d_1 , d_2 , d_3 , i d_4 . El procés és totalment anàleg a extreure les xifres d'un nombre en base 10, però en base 64. El que estem fent és calcular els dígit d_i de la fórmula següent:

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

(Cal recordar que el procés d'extracció de xifres treballat a PRO1 produeix les xifres al revés, és a dir, començant per d_4 .)

3. Per últim, usant la funció `base64_to_char` convertim d_1 , d_2 , d_3 i d_4 en caràcters i els mostrem a la sortida en aquest ordre.

Si la seqüència d'entrada no té un llargada múltiple de 3, just al final, tindrem un grup de només 1 o 2 bytes:

- Si tenim un grup de 2 bytes: assignem $B_3 = 0$, i seguim igualment els passos de la codificació. Un cop amb les xifres d_i , canviem d_4 pel caràcter '='.
- Si tenim un grup d'1 byte: assignem $B_2 = 0$ i $B_3 = 0$, i seguim igualment els passos de la codificació. Un cop amb les xifres d_i , canviem tant d_3 com d_4 pel caràcter '='.

Entrada

L'entrada consisteix en diverses seqüències de bytes. Cadascuna comença amb un enter n que indica el nombre de bytes que segueixen i després hi ha n bytes, a on cadascun és només un nombre natural entre 0 i 255 (ambdós inclosos).

Sortida

La sortida ha de ser una línia per a cada cas amb la codificació en base 64 de la seqüència de bytes de l'entrada, sense espais entre els caràcters.

Observacions

- Aquest problema té com a centre d'interès la **correctesa** i la **llegibilitat**. Sobre la llegibilitat, es valorarà que el programa utilitzi funcions per evitar repetició i separar les diverses tasques.
- Si teniu la funció `base64_to_char` i el Jutge us l'ha acceptat, useu-la directament. Si no, copieu la següent definició:

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

Exemple d'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
```

Exemple d'entrada 2

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

Exemple d'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
```

Exemple de sortida 1

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

Exemple de sortida 2

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

Exemple de sortida 3

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

Exemple d'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
```

Exemple de sortida 4

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

Informació del problema

Autor : Pau Fernández

Generació : 2025-10-30 12:08:22

© *Jutge.org*, 2006–2025.

<https://jutge.org>