

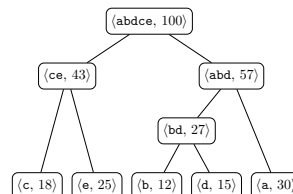
1. El peso de este ejercicio en la nota del examen de la práctica es de un 33.33% (1/3 de la nota).
2. La evaluación es completamente automática (nota manual: 0%, nota automática: 100%)
3. El peso de los juegos de pruebas público y privado en el cálculo de la nota automática es idéntico (público: 5/10, privado: 5/10).

Dado un árbol de códigos  $t$  y un string  $s$  de 0's y 1's queremos una función

```
int es_codigo(const BinTree< pair<string,int> >& t, const string& s);
```

que devuelve 0 si el string  $s$  es el código de algún símbolo en el "treecode"  $t$ , devuelve -1 si el string  $s$  es prefijo del código de algún símbolo en  $t$  pero  $s$  no es un código, y devuelve 1 si existe un código en  $t$  que es un prefijo de  $s$  pero  $s$  no es un código. Informalmente, se devuelve -1 si  $s$  "es demasiado corto", 1 si  $s$  "es demasiado largo" y 0 si es un código en el treecode  $t$ .

Así por ejemplo, para el árbol de la figura los símbolos codificados son a, b, c, d, e y sus códigos respectivos son 11, 100, 00, 101 y 01. La función devolverá 0 si  $s = "00"$ , devolverá -1 si  $s = "10"$  y devolverá 1 si  $s = "111"$ .



Se os suministra un módulo (ficheros `treecodeIO.cc` y `treecodeIO.hh`) con las operaciones de entrada/salida de "treecodes". Con todo ello escribiréis un pequeño programa que lee una serie de casos, cada caso formado por un "treecode" y una secuencia de strings binarios e imprime, para cada caso, si los strings son códigos o no.

## Entrada

La entrada consiste en una serie de casos. Para cada caso se da una secuencia en formato válido que representa a un "treecode" (se puede leer usando la función `leer_treecode` que os proporcionamos); si el treecode no es vacío después del "treecode" viene un valor entero  $k \geq 0$  y a continuación una secuencia de  $k$  strings binarios (esto es, sólo contienen 0s y 1s).

La secuencia de casos termina con un "treecode" vacío, siendo todos los casos precedentes "treecodes" que codifican 2 símbolos o más.

## Salida

Para cada caso, si el "treecode" leído no es vacío se imprime el valor  $k$  seguido por  $k$  pares (string,entero), formado cada uno por el correspondiente string  $s$  dado seguido del valor (-1,0 o 1) retornado por la función `es_codigo` para el string  $s$ .

La salida de cada caso acaba con un salto de línea.

## Ejemplo de entrada

```
abdce 100 ce 43 c 18 -1 -1 -1 -1 e 25 -1 -1 -1 -1
abd 57 bd 27 b 12 -1 -1 -1 -1 d 15 -1 -1 -1 -1
a 30 -1 -1 -1 -1
4 00 101 1 110

R 36 A 16 B 8 e 4 -1 -1 -1 -1 C 4 n 2 -1 -1 -1 -1
D 2 o 1 -1 -1 -1 -1 u 1 -1 -1 -1 -1 E 8 a 4 -1 -1
-1 -1 F 4 t 2 -1 -1 -1 -1 m 2 -1 -1 -1 -1
G 20 H 8 I 4 i 2 -1 -1 -1 -1 J 2 x 1 -1 -1 -1 -1
p 1 -1 -1 -1 -1 K 4 h 2 -1 -1 -1 -1 s 2 -1 -1 -1 -1
L 12 M 5 N 2 r 1 -1 -1 -1 -1 l 1 -1 -1 -1 -1 f 3
-1 -1 -1 -1 spc 7 -1 -1 -1 -1
8 0110 00110 101 1111 000 10010 10111 0011
-1 -1
```

## Ejemplo de salida

```
4 00 0 101 0 1 -1 110 1
8 0110 0 00110 0 101 -1 1111 1 000 0 10010 0 10111 1 0011 -1
```

## Información del problema

Autor : Profesores de PRO2  
Generación : 2019-05-31 19:44:01

© *Jutge.org*, 2006–2019.  
<https://jutge.org>