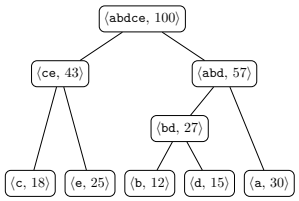
Examen práctica - Turno 2 - Problema 1 - Es código? X34462_es

1. El peso de este ejercicio en la nota del exámen 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 arbol de códigos t y un string s de 0's y 1's queremos un procedimiento

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

que retorna con `res == -1` si s un prefijo del código de algún símbolo en el "treecode" t pero no un código, retorna con `res == 0` si s es el código de algún símbolo en t, y retorna con `res == 1` si existe un código en t que es un prefijo de s pero s no es un código. Además, si `res == 0` entonces en `codif` se devuelve el símbolo codificado por s, y en caso contrario se retorna con `codif == "**"`. Informalmente, `res==0` si s es un código en t, `res==-1` si s es "demasiado corto" y `res==1` si s es "demasiado largo".



Por ejemplo, en el árbol t de la figura los símbolos codificados son a, b, c, d, e y sus códigos respectivos son `11`, `100`, `00`, `101` y `01`. Por lo tanto `es_codigo(t, s, res, codif)` retornará con `res==0` y `codif=="c"` si $s =$ `"00"`. Si $s =$ `"1001"` entonces el procedimiento retornará con `res==1` y si $s =$ `"10"` entonces retornará con `res==-1`. En estos dos últimos casos `codif == "**"` ya que en ambos tenemos `res != 0`).
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, y cúal es el símbolo codificado, en su caso.

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 de *k* ternas, formada cada una por el correspondiente string *s* dado, seguido del resultado `res` (-1, 0 o 1) y el valor del string `codif` (es decir, el símbolo codificado por *s* si `res == 0` o `"**"` si `res != 0`).
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 c 101 0 d 1 -1 ** 110 1 **
8 0110 0 t 00110 0 o 101 -1 ** 1111 1 ** 000 0 e 10010 0 x 10111 1 ** 0011 -1 **
```

Información del problema

Autor : Profesores de PRO2
Generación : 2019-05-31 17:29:06