Universitat Politècnica de Catalunya
Facultat d'Informàtica de Barcelona

Cognoms, Nom D.N.I.

Titulació: Grau en Enginyeria Informàtica **Curs:** Q2 2020–2021 (1r Parcial)
Assignatura: Programació 2 (PRO2) **Data:** 19 d'abril de 2021
Duració: 1h 30m

1. **(5 puntos)** Dado un árbol binario (`BinTree<int>`) no vacío a y un nodo cualquiera x en a, diremos que el camino $P(a, x)$ de la raíz de a a x es la secuencia de nodos n_0, n_1,..., n_k, tal que n_0 es la raíz de a, $n_k = x$ y para toda i, $0 \leq i < k$, n_i es el padre del nodo n_{i+1}. Observad que fijado un cierto nodo x, el camino $P(a, x)$ es único. El valor $\text{val}(P)$ de un camino P es la suma de los valores en los nodos del camino. El valor máximo m de los caminos de un árbol a es entonces

$$m = \max\{\text{val}(P(a, x)) \,|\, x \text{ es un nodo en } a\}$$

Un camino del árbol se puede representar mediante una `list<char>`, formada por caracteres `'L'` o `'R'` de la siguiente forma: si el camino es la secuencia de nodos n_0,n_1,...,n_k entonces se representará por la secuencia de caracteres c_1,...,c_k, donde si n_i es hijo izquierdo de n_{i-1} entonces $c_i =$ `'L'`, y si n_i es el hijo derecho de n_{i-1} entonces será $c_i =$ `'R'`.

Por ejemplo, si a es el siguiente árbol (que sólo contiene enteros no negativos)

```
        4
      /   \
     /     \
    3       2
   / \     / \
  1   0   4   4
     /       / \
    3       0   0
```

entonces el valor máximo de un camino de a es $m = 10$ y existen cinco caminos con dicho valor máximo, representados por las siguientes listas de caracteres: ['L','R','L'], ['R','L'], ['R','R'], ['R','R','L'] y ['R','R','R'].

Observad que, si todos los valores en a son enteros no negativos, siempre existirá un camino desde la raíz a una hoja cuyo valor sea el máximo m. En el ejemplo, cuatro de los cinco caminos de valor máximo de a llegan hasta las hojas.

Se pide:

(a) (2.5 puntos) Completa el diseño de la siguiente función recursiva:

// Pre: a es un árbol no vacío que sólo contiene enteros no negativos
// Post: el resultado es el valor máximo de los caminos del árbol a

```
int maxvalorcamino(const BinTree<int>& a) {
    int v = a.value();
    BinTree<int> e = a.left();
    BinTree<int> d = a.right();
    ...
}
```

(b) (2.5 puntos) Justifica la corrección de la función diseñada en el apartado anterior, incluyendo la demostración de que termina siempre.

SOLUCIÓ:

Cognoms, Nom D.N.I.

2. **(5 puntos)** Diseña un procedimiento `shuffle` que recibe dos listas de enteros `l1` y `l2` y un entero $k > 0$ y modifica la lista `l1` de manera que los elementos de `l2` aparecen intercalados cada `k` elementos de `l1` empezando desde el principio. Asumiremos que la lista `l1` tiene $m \geq n \cdot k$ elementos, siendo n el número de elementos de `l2`. El procedimiento deja vacía la lista `l2`.

   Por ejemplo si `l1` $= [3, 5, 7, 1, 2, 8, 9, 4, 6]$, `l2` $= [200, 300, 100]$ y `k` $= 2$ entonces despues de la llamada `shuffle(l1, l2, k)` tendremos `l2` $= []$ y

   $$\texttt{l1} = [3, 5, 200, 7, 1, 300, 2, 8, 100, 9, 4, 6]$$

   Supongamos que definimos la siguiente función abstracta SHUFFLE (que no es usable como parte de ningún código):

   $$\begin{aligned}&\mathrm{SHUFFLE}([x_1, \ldots, x_m], [y_1, \ldots, y_n], k)\\ &\qquad = [x_1, \ldots, x_k, y_1, x_{k+1}, \ldots, x_{2k}, y_2, \ldots, x_{nk}, y_n, x_{nk+1}, \ldots, x_m]\end{aligned}$$

   Entonces nuestro procedimiento `shuffle` puede especificarse asi:

```
// Pre: l1 = L1, l2 = L2, k > 0, |L1| >= |L2|·k
// Post: l1 = SHUFFLE(L1, L2, k), l2 = []
   void shuffle(list<int>& l1, list<int>& l2, int k);
```

   Utiliza esta plantilla para tu código C++

```
void shuffle(list<int>& l1, list<int>& l2, int k) {
   list<int>::iterator it = ...

   // Inv: l1 = ... ,
   //      L2 = L2'·l2  (esto es, L2' es la sublista prefijo de L2
   //           que no esta en la lista l2),
   //      it referencia a ...
   while ( ... ) {
      for (int j = 0; ... ; ...) {
          ...
      }
      ...
   }
}
```

   Se pide:

   (a) (2.5 puntos) Completa la implementación del procedimiento `shuffle`, utilizando la plantilla que se proporciona.

(b) (1.5 puntos) Completa el invariante del bucle `while` que se proporciona y justifica la corrección del procedimiento `shuffle`, excepto su terminación (véase el apartado siguiente).

(c) (1 punto) Escribe una función de cota para el bucle `while` y justifica que `shuffle` siempre termina.

SOLUCIÓ: