Universitat Politècnica de Catalunya
Facultat d'Informàtica de Barcelona

Titulació: Grau en Enginyeria Informàtica
Assignatura: Programació 2 (PRO2)
Duració: 2h 30m

Curs: Q2 2019–2020 (1r Parcial)
Data: 14 de novembre de 2019

1. **(5 punts)** Volem aplicar una sèrie d'altes, baixes i modificacions a una llista ordenada de `pair<string, int>`.

   Diem que una *transacció* és una tripleta `< <string, int>, char>`. El component `char` pot valer `'A'`, `'B'` o `'M'`, que signifiquen respectivament `'alta'`, `'baixa'` i `'modificació'`.

   Diem que s'ha aplicat una transacció $t = \langle\langle \texttt{s}, i\rangle, \texttt{op}\rangle$ a una llista `l` de `pair<string, int>` quan:

   - si `op = 'A'`: si `s` no hi era a `l`, s'ha afegit el parell $\langle \texttt{s}, i\rangle$ a `l`; en cas contrari no s'ha fet res
   - si `op = 'B'`: si `s` hi era a `l`, s'ha esborrat d'`l` el parell d'`s` a `l`; en cas contrari no s'ha fet res
   - si `op = 'M'`: si `s` hi era a `l`, s'ha substituït el company d'`s` a `l` per `i`; en cas contrari no s'ha fet res

   Considereu la següent operació que rep una llista `l` de `pair<string, int>` i una llista `lt` de transaccions, on cada una està ordenada de manera estrictament creixent per la string (no conté strings repetides). L'operació aplica a `l` totes les transaccions de `lt`.

```
typedef pair<string,int> parell; // els camps es diuen first i second

struct transaccio {
  parell dades;
  char operacio; // possibles valors 'A', 'B' o 'M'
};

// Pre: l = L; l i lt estan ordenades de manera estrictament creixent
        per la string (no conté strings repetides)
// Post: l és el resultat d'aplicar a L totes les transaccions d'lt

void aplica_transaccions(list<parell>& l, const list<transaccio>& lt);
```

   Exemple: siguin `l` i `lt`

   $l = [\langle \texttt{bleda}, 12\rangle, \langle \texttt{ceba}, 7\rangle, \langle \texttt{col}, 8\rangle, \langle \texttt{enciam}, 11\rangle, \langle \texttt{espinacs}, 15\rangle]$
   $lt = [\langle\langle \texttt{bleda}, 0\rangle, \texttt{'B'}\rangle, \langle\langle \texttt{ceba}, 6\rangle, \texttt{'M'}\rangle, \langle\langle \texttt{escarola}, 18\rangle, \texttt{'A'}\rangle, \langle\langle \texttt{espinacs}, 0\rangle, \texttt{'B'}\rangle, \langle\langle \texttt{tomaquet}, 18\rangle, \texttt{'A'}\rangle]$

   Després d'aplicar l'operació, `l` ha de quedar

   $l = [\langle \texttt{ceba}, 6\rangle, \langle \texttt{col}, 8\rangle, \langle \texttt{enciam}, 11\rangle, \langle \texttt{escarola}, 18\rangle, \langle \texttt{tomaquet}, 18\rangle]$

   Volem que feu el següent:

   (a) (2 punts → 3 punts) Escriviu el codi per a aquesta operació
   (b) (2 punts → 1 punt) Escriviu l'invariant de tots els bucles que heu dissenyat
   (c) (1 punt) Doneu la funció de fita dels bucles i justifiqueu que acaban sempre

   Valorarem que les vostres solucions siguin eficients en temps i espai.

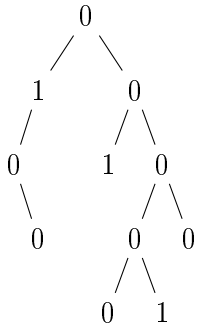
2. **(5 punts)** En aquest exercici heu de trobar solucions recursives pels dos problemes plantejats en els dos primers apartats, i al tercer apartat heu de justificar la correcció d'una de les dues solucions trobades. La funció que us demanem a l'apartat (b) l'heu d'especificar vosaltres i també tota operació nova que feu servir, tant a l'apartat (a) com a l'apartat (b). Noteu que algunes de les vostres solucions poden beneficiar-se d'un bon disseny recursiu per immersió, per exemple, per evitar còpies i assignacions de vectors.

   (a) (2 punts) Donat un arbre binari `a` d'enters que conté només 0s i 1s, volem una funció que obtingui el nombre de 0s i el nombre de 1s que conté `a`.

   Feu servir la següent especificació

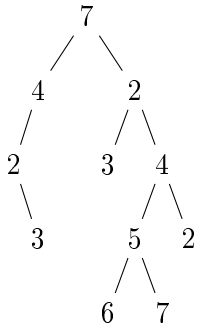
   ```
   // Pre: tots els elements d'a són 0s o 1s
   // Post: zeros = nombre de 0s que conté l'arbre a,
   //       uns = nombre d'1s que conté l'arbre a

   void freq_zeros_uns(const BinTree<int>& a, int& zeros, int& uns);
   ```

   Per exemple, amb l'arbre

   l'operació hauria d'obtenir que hi ha 8 vegades el número 0 i 3 vegades el número 1.

   (b) (2 punts) Ara volem una funció que donat un arbre binari `a` d'enters entre 0 i 9, obtingui les freqüències de cada element, és a dir, quants 0s, 1s, 2s, ... conté l'arbre. Per exemple, amb l'arbre

   l'operació hauria d'obtenir que no hi ha cap 0, no hi ha cap 1, que hi ha 3 vegades el número 2, 2 vegades el número 3, 2 vegades el número 4, etc. Especifiqueu i implementeu la funció demanada.

   (c) (1 punt) Justifiqueu la correcció de la solució que heu dissenyat per l'apartat (a) o per l'apartat (b) —una de les dues, la que vulgueu.

Valorarem que les vostres solucions siguin eficients en temps i espai.