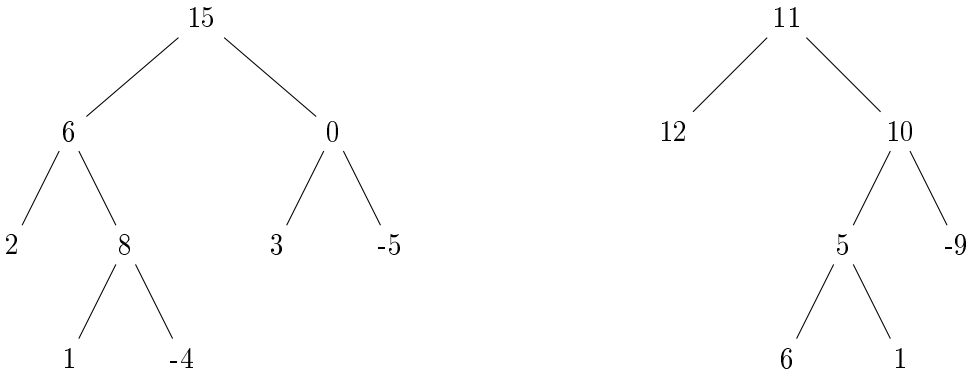
Parcial Programació 2

Enginyeria Informàtica
Novembre 2015
Temps estimat: 2h 30m

Problema 1 (5 punts)

En un arbre binari d'enters, diem que un node és *dominador* si conté un valor més gran que tots els dels nodes que són descendents seus. Per exemple, en l'arbre de l'esquerra són dominadors els nodes amb valors 15, 2, 8, 1, -4, 3 i -5; en l'arbre de la dreta, són dominadors els nodes amb valors 12, 10, 6, 1 i -9.



Fixeu-vos que, per definició, tota fulla és un node dominador perquè no té cap descendent. L'arbre buit té 0 elements dominadors.

Volem donar en els apartats següents una implementació eficient de la funció:

```
int nombre_dominadors(Arbre<int>& a);
/* Pre: a = A i tot node d'A te 0 o 2 fills */
/* Post: el resultat és el nombre de nodes dominadors d'A */
```

Problema 1.1 (1 punt)

Suposeu que un arbre A té un valor x a l'arrel i arbres $A1$ i $A2$ com a fills esquerre i dret. Digueu, en paraules (no codi), quants elements dominadors té A en funció del nombre d'elements dominadors d'$A1$ i $A2$, de x i, potser, d'altres quantitats que són funcions d'$A1$ i $A2$.

Problema 1.2 (1.5 punts)

Amb aquesta definició al cap, doneu la capçalera d'una funció d'immersió

```
int i_nombre_dominadors(Arbre<int>& a, parametres nous);
/* Pre: a = A i tot node d'A te 0 o 2 fills (... i, si cal, propietats dels parametres nous) */
/* Post: el resultat retornat és el nombre de nodes dominadors d'A
         (... i, si cal, propietats dels parametres nous) */
```

que permeti una implementació eficient de `nombre_dominadors`, i completeu la seva especificació.

Problema 1.3 (0.2 punts)

Doneu una implementació de `nombre_dominadors` fent servir `i_nombre_dominadors`.

Problema 1.4 (2.3 punts)

Doneu la implementació de `i_nombre_dominadors`. Cal que el cost d'aquesta implementació sigui *lineal* en la mida de l'arbre `a`. Si no és així, potser la funció d'implementació que heu proposat no és adequada i haureu de revisar els apartats anteriors.

Si feu servir alguna altra funció auxiliar, cal que també l'especifiqueu i implementeu.

Noteu que en aquest problema no se us demana que justifiqueu les vostres implementacions.

Problema 2 (5 punts)

Volem dissenyar un procediment que, donat un `vector<int> v` amb els dos primers elements diferents, trobi els elements que són màxims locals i els emmagatzemi en una cua de parells d'enters que representen la posició en el vector i el valor corresponent.

Per a qualsevol posició `k` entre `1` i `v.size()-2`, direm que `v[k]` és un màxim local a la posició `k` si i només si `v[k]>v[k+1]` i existeix una posició `j`, `1<=j<=k`, tal que `v[j]>v[j-1]` i tots els elements del subvector `v[j..k]` són iguals (aquesta última condició es satisfà, en particular, si el subvector és d'un únic element `v[k..k]=v[k]`). Noteu que, per definició, ni `v[0]` ni `v[v.size()-1]` seran mai màxims locals.

En tot el problema, "ordenat" s'ha d'entendre "ordenat creixentment per posició".

La capçalera del programa desitjat és:

```
void obtenir_maxims_locals(const vector<int> &v, queue<pair<int,int>> &maxims);
// Pre: v.size() >= 2, v[1] != v[0], maxims és una cua buida
// Post: maxims conté els parells (posició,valor) dels màxims locals de v, ordenats
```

Com a exemple, si el vector `v` conté els següents 16 elements (des de la posició 0 fins a la 15):

-5 -1 2 2 4 5 5 5 3 6 11 7 -2 -2 0 3

la cua `maxims` ha de contenir al final els parells (7, 5) (10, 11).

Per dissenyar el procediment proposem usar un bucle amb una variable entera `k` més una o més variables que podeu declarar lliurement, però satisfent el següent invariant parcial:

```
1 <= k < v.size(),
maxims conté els parells (posició,valor) dels màxims locals de v[0..k-1], ordenats
```

Problema 2.1 (3 punts)

Feu un algorisme iteratiu (obligatòriament amb `while`, no useu `for`) per al problema que respecti aquest invariant parcial, afegint les variables locals que us calguin.

El temps d'execució de l'algorisme hauria de ser lineal en `v.size()`, i es valorarà que l'algorisme no faci comparacions redundants.

Problema 2.2 (2 punts)

Escriviu l'invariant complet del vostre bucle, afegint a l'invariant parcial donat les condicions que calguin sobre les noves variables que declareu i useu vosaltres.

I llavors justifiqueu que:

1. Les vostres inicialitzacions estableixen l'invariant.
2. El cos del vostre bucle manté l'invariant.
3. En acabar el bucle, per l'invariant, se satisfà la Post.

Encara que no és necessari usar cap nova operació auxiliar per resoldre el problema, si declareu qualsevol altre procediment o funció, l'haureu d'haver especificat i en aquest apartat cal que justifiqueu la correctesa de la seva implementació.