Parcial Programació 2

Enginyeria Informàtica
Maig 2011
Temps estimat: 2h 20m

Problema 1: *Comptatge d'elements solitaris d'un vector* (50%)
Donat un vector d'elements de qualsevol tipus dotat d'operador d'igualtat, diem que un element és *solitari* si i només si és diferent dels seus elements anterior i següent al vector (dit d'una altra manera, si no forma una parella de valors consecutius iguals ni amb l'anterior ni amb el següent). Per definició, diem que el primer element és sempre diferent del seu anterior (tot i que no existeix) i que l'últim element és sempre diferent del seu següent (tot i que no existeix).

Es demana dissenyar una funció per comptar el nombre de solitaris en un vector d'enters.

Per exemple, el vector:

```
 -7   5   5   5  -2   2   5   1   1
```

conté 4 elements solitaris: el primer (-7), el cinquè (-2), el sisè (2) i el setè (5).

El disseny ha de seguir el següent esquema iteratiu, de manera que el codi compleixi estrictament l'invariant del bucle que us donem.

```
int solitaris(const vector<int> &v)
/* Pre: v.size() > 0 */
/* Post: el resultat és el nombre de solitaris a v */
{
  // Aquí han d'anar les declaracions de les variables locals
  // i totes les inicialitzacions que calguin per satisfer l'invariant
  ...

  /* Inv: n = nombre de solitaris en v[0..i-2], 1 <= i <= v.size(),
          b indica si v[i-1] és diferent del seu element anterior */

  // Cal proporcionar la condició del bucle
  while ( ... ) {
    // Cal dissenyar el cos del bucle
    ...

  }
  /* A:   n = nombre de solitaris en v[0..v.size()-2],
          b indica si v[v.size()-1] és diferent del seu element anterior */

  // Cas que calgui més codi per complir la Post, s'ha d'escriure aquí
  ...

  return n;
}
```

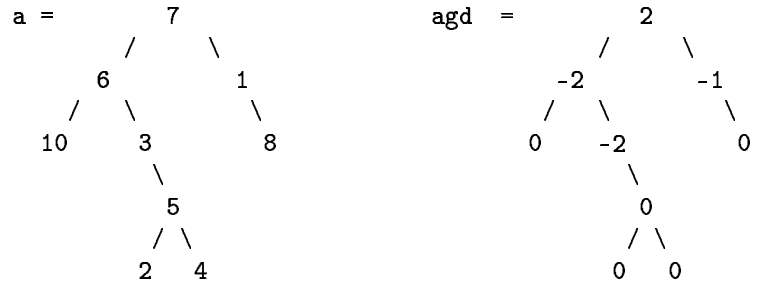
Al disseny heu d'incloure la **justificació** de tots els elements: inicialitzacions, condició de sortida, cos del bucle i acabament.

Es valorarà **molt negativament** fer recorreguts innecessaris del vector.

Problema 2: *Arbre de graus de desequilibri* (50%)
Considereu un arbre binari a de qualsevol tipus. Donat qualsevol subarbre b no buit d'a, diem que el *grau de desequilibri* de b és la diferència entre l'altura del seu fill esquerre i la del seu fill dret (pot ser positiva, zero o negativa).

Diem que l'*arbre de graus de desequilibri* d'a és un altre arbre binari *agd* amb la mateixa forma que a, on cada element és el grau de desequilibri del subarbre d'a que té com a arrel el corresponent de l'esmentat element.

Exemple:



Dissenyeu una funció que retorni l'arbre de graus de desequilibri d'un arbre binari d'enters. Podeu fer servir la següent especificació:

```
Arbre<int>  arbre_graus_desequilibri (Arbre<int> &a)
/* Pre: a=A */
/* Post: el resultat és un arbre amb la mateixa estructura que A on cada
   node conté el grau de desequilibri del subarbre d'A corresponent */
```

Al disseny heu d'incloure la **justificació** de tots els elements de la funció i de les seves auxiliars que siguin recursives, si en té: casos base i recursius, validesa de les crides i acabament o decreixement.

Es valorarà **molt** l'eficiència de la solució proposada. Per tant, heu d'evitar la repetició de càlculs amb immersions d'eficiència.

Us recordem l'especificació de les operacions que necessiteu de la classe Arbre.

```
Arbre() // Constructora; s'executa automàticament al fer una declaració Arbre a;
/* Pre: cert */
/* Post: El resultat és un arbre buit */

void plantar(const T& x, Arbre& a1, Arbre& a2)
/* Pre: El paràmetre implícit és buit, a1 = A1, a2 = A2 */
/* Post: El paràmetre implícit té x com a arrel, A1 com a fill esquerre i A2
   com a fill dret; a1 i a2 són buits  */

void fills(Arbre& fe, Arbre& fd)
/* Pre: El paràmetre implícit no està buit i li diem A; fe i fd són buits */
/* Post: fe és el fill esquerre d'A; fd és el fill dret d'A; el paràmetre
   implícit és buit */

T arrel() const
/* Pre: El paràmetre implícit no és buit */
/* Post: El resultat és l'arrel del paràmetre implícit */

bool es_buit() const
/* Pre: cert */
/* Post: El resultat indica si el paràmetre implícit és buit o no */
```