Universitat Politècnica de Catalunya
Facultat d'Informàtica de Barcelona

Titulació: Grau en Enginyeria Informàtica
Assignatura: Programació 2 (PRO2)
Duració: 1h 30m

Curs: Q1 2020–2021 (1r Parcial)
Data: 5 de novembre de 2020

1. **(5 punts)** Considereu la següent acció que modifica un vector d'enters v reordenant els seus elements de manera que els negatius queden en un subvector a l'esquerra, els zeros en un subvector al mig, i els positius en un subvector a la dreta. És irrellevant en quin ordre queden els elements internament tant en el subvector de negatius com en el subvector de positius. L'acció té dos paràmetres enters de sortida a i b que marquen la frontera entre els tres subvectors. Noteu però què, depenent del contingut de v, un o dos d'aquests subvectors podria ser buit.

```
// Pre: v = V
// Post: v és una permutació de V, 0 ≤ a ≤ b ≤ v.size(),
//       tots els elements de v[0...a-1] són negatius,
//       tots els elements de v[a...b-1] són zero,
//       tots els elements de v[b...v.size()-1] són positius

void reordena(vector<int>& v, int& a, int& b) {
    a = ____________;
    b = ____________;
    int c = ____________;
    while (____________) {
        if (____________) {
            ____________
            ____________
            ____________
        }
        else if (____________) {
            ____________
            ____________
            ____________
        }
        else {
            ____________
            ____________
            ____________
        }
    }
}
```

Dins del codi de `reordena` no es pot cridar a l'operació `sort`, però sí es pot cridar a l'operació auxiliar `swap`:

```
// Pre: v = V, 0 ≤ i, j ≤ v.size() − 1, V[i] = X, V[j] = Y
// Post: v[i] = Y, v[j] = X, i per a les altres posicions diferents k
//       entre 0 i v.size() − 1 tenim que v[k] = V[k]
void swap(vector<int>& v, int i, int j);
```

Us proposem el següent invariant **incomplet**, el qual introdueix la variable local entera c:

```
// Inv: v és una permutació de V,
//      tots els elements de v[0...a − 1] són negatius,
//      tots els elements de v[a...b − 1] són zero,
//      tots els elements de v[c...v.size() − 1] són positius
```

Donat aquest invariant, penseu en el significat del subvector $v[b \ldots c-1]$ i en quin hauria de ser el seu estat inicial (abans d'entrar en el bucle) i el seu estat final (després de sortir del bucle). Llavors,

a) (0,5 punts) Completeu l'invariant proposat per al bucle de `reordena` escrivint només les condicions que cal afegir.

b) (0,5 punts) Doneu una funció de fita.

c) (2,5 punts) Ompliu el codi faltant (només els llocs indicats per les capses). Cada capsa s'ha d'omplir amb una expressió o amb una o més instruccions.

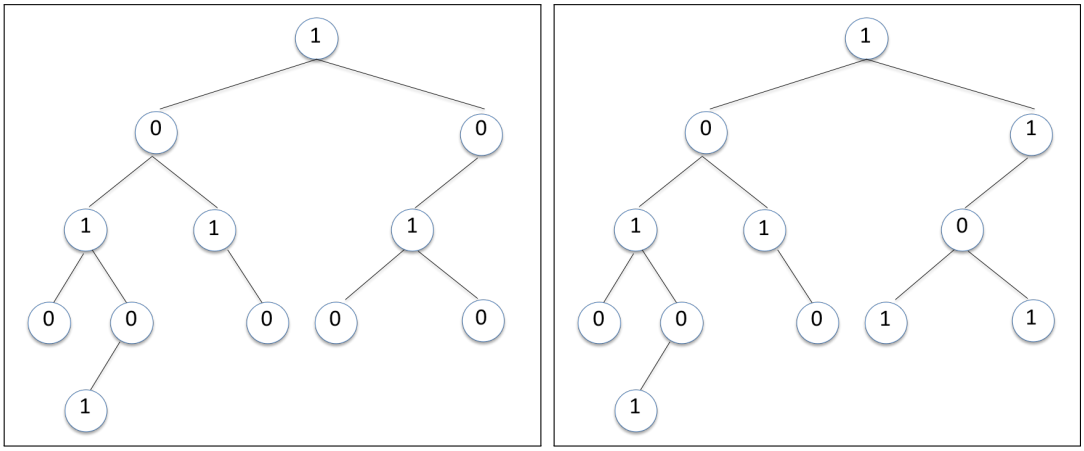
d) (1,5 punts) Justifiqueu la correctesa del codi, incloent les inicialitzacions, la condició del bucle, el cos del bucle i per què acabarà sempre (usant l'invariant i la funció de fita donats).

2. **(5 punts)** Un arbre 0-1 és un arbre binari amb valors que són només zeros i uns, que compleix que, per a tot node n, si el valor a n és 0 aleshores el valor a les arrels del seu fill dret i del seu fill esquerre (si existeixen) és 1 i, recíprocament, si el valor a n és 1 aleshores el valor a les arrels del seu fill dret i del seu fill esquerre (si existeixen) és 0.

   Es demana escriure i justificar la correcció d'una funció recursiva que donat un arbre binari d'enters ens digui si és un arbre 0-1. L'especificació Pre/Post de la funció és:

```
// Pre: a conte nomes zeros i uns
bool arbre01(const BinTree<int> &a);
// Post: El resultat ens diu si a es un arbre 0-1
```

   Per exemple, l'arbre a l'esquerra és un arbre 0-1, però l'arbre a la dreta no ho és pas:



   a) (2,5 punts) Implementa la funció `arbre01`.

   b) (2,5 punts) Justifica la correcció i acabament de la teva implementació de la funció `arbre01`.