Universitat Politècnica de Catalunya
Facultat d'Informàtica de Barcelona

Cognoms, Nom | D.N.I.

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

Curs: Q2 2020–2021 (2n Parcial)
Data: 14 de juny de 2021

1. **(5 punts)** Donada la següent representació d'una classe de llistes amb punt d'interès, doblement encadenades i sense sentinella

```
class Llista {
  private:
    struct node_llista {
        int info;
        node_llista* seg;
        node_llista* ant;
    };
    int longitud;
    node_llista* primer_node;
    node_llista* ultim_node;
    node_llista* act;
    ... // especificació i implementació d'operacions privades

  public:
    ... // especificació i implementació d'operacions públiques
};
```

es demana:

a) Implementar el mètode `elimina2` amb la següent especificació:

```
void elimina2();
/* Pre: La llista implícita conté almenys dos elements;  el punt
   d'interès apunta un element de la llista que té un successor
   (és a dir, no apunta l'últim) */
/* Post: S'ha eliminat de la llista ímplicita el punt d'interès i
   el seu successor; a la llista resultant el punt d'interès passa a
   ser el successor del successor del punt d'interès original */
```

b) Implementar el mètode `elimina_parell_zero` amb la següent especificació:

```
void elimina_parell_zero();
/* Pre: La llista implícita és una llista L no buida */
/* Post: La llista ímplicita és el resultar d'eliminar d'L tots els
   parells d'elements consecutius que sigui necessari de manera que no
   hi hagi cap parell d'elements consecutius la suma dels quals sigui 0 */
```

Per exemple, si `L` $= [12, 5, -5, -7, 8, 45, -45, -8, 7, 4]$ llavors després d'executar `L.elimina_parell_zero()` tindrem `L` $= [12, 4]$. Fixeu-vos que després d'eliminar 45 i -45, també haurem d'eliminar 8 i -8, i 7 i -7, altrament el resultat contindria un parell d'elements consecutius la suma dels quals és 0.

Implementeu aquesta operació utilitzant `elimina2` com a funció auxiliar.

Apart de l'ús d'`elimina2` per a implementar `elimina_parell_zero`, el codi d'aquestes dues operacions no ha de cridar a cap altre mètode públic o privat de la classe `Llista`, ni de cap altra classe, només consultar i modificar els atributs i els nodes de la llista implícita.

Per a la vostra solució ompliu el codi que falta a les capses indicades. Cada capsa ha de contenir exactament una instrucció simple o una expressió. S'ha de respectar l'invariant del bucle.

SOLUCIÓ:

a)

```
void Llista::elimina2(){
    nodo_lista* aux = act;
    act = act -> seg -> seg;
    if (                          ){
        primer = ultim = nullptr;
    } else if (primer == aux){
        primer =            ;
                                  ;
    } else if (                   ) {
        ultim =            ;
                                  ;
    } else {
                                  ;
                                  ;
    }
    delete            ; delete            ;
    longitud -= 2;
}
```

b)

```
void Llista::elimina_parell_zero(){
    act = primer;
    nodo_lista* p = [          ];
    // Inv: no hi ha dos nodes consecutius
    //   que sumin 0 abans del node apuntat per p,
    //   si act != nullptr llavors p == act -> seg
    while ([          ]) {
        if ([          ]) {
            [          ];
            if (act == nullptr)
              // hem acabat
              p = nullptr;
            else if ([          ])
              // si act apunta al primer de la lista
              [          ];
            else {
              [          ];
              [          ];
            }
        } else {
            [          ];
            [          ];
        }
    }
}
```

2. **(5 punts)** Donada la següent definició d'una classe `ArbreNari` en C++

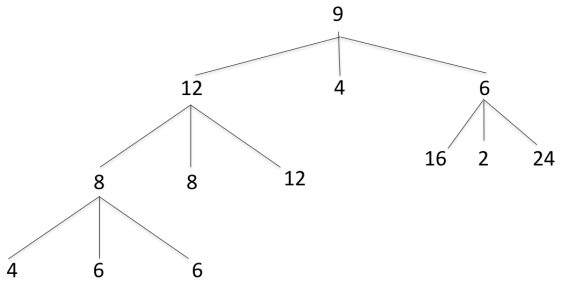
```
class ArbreNari{
  private:
    struct node {
        int info;
        vector<node*> fill;
    };
    node* arrel; // apuntador a l'arrel de l'arbre
    int N; // aritat de l'arbre
    ...
  public:
    ...
};
```

es demana implementar:

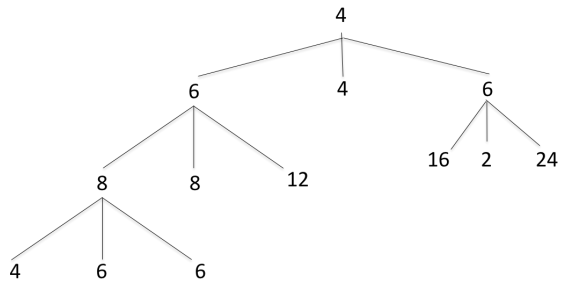
a) Un mètode privat `es_fulla` que torni cert si i només si el node apuntat per un apuntador p donat és una fulla (té tots els fills buits)

b) Un mètode públic `mult` que torni cert si i només si per a tots els subarbres de l'arbre implícit es compleix la següent propietat: la suma dels valors del subarbre és un múltiple del nombre de nodes del subarbre.

Per exemple, l'arbre de la figura té la propietat a comprovar mitjançant `mult` i s'hauria de retornar `true`



Per contra, l'arbre de la següent figura no compleix la propietat; d'una banda el propi arbre té com a suma dels seus valors 106 i la seva talla és 13; d'altra banda els valors en el primer fill sumen 50, que no és múltiple de 7.



Les vostres implementacions han de ser eficients, sense càlculs inútils o repetits. Per implementar `mult` haureu de completar i fer servir aquesta immersió:

```
  static bool ArbreNari::i_mult(node* p, ...);
/* Pre: p és un punter al node arrel d'un arbre N-ari no buit */
/* Post: retorna true si i només si tots els subarbres de l'arbre
   amb node arrel apuntat per p compleixen que la suma dels seus valors és
   múltiple del seu nombre de nodes.
   A més, si la funció retorna true, llavors ... */
```

Ompliu les capses amb el vostre codi, respectant la resta del codi ja escrit; completeu també l'especificació del mètode `i_mult`. Cada capsa pot contenir ara més d'una instrucció.

N.B.: Tingueu molt present que la divisió $0/0$ és un error.

SOLUCIÓ:

a)

```
/* Pre: cert */
/* Post: retorna true si i només si p apunta a l'arrel d'un arbre
   N-ari consistent en un únic node */
static bool ArbreNari::es_fulla(node* p) {
  // Escriu aquí la teva implementació
}
```

b)

```
/* Pre: l'arbre N-ari implícit no és buit */
/* Post: retorna true si i només si tots els subarbres de l'arbre
   implícit compleixen que la suma dels seus valors és múltiple
   del seu nombre de nodes */
bool ArbreNari::mult( ) const {
  // Escriu aquí la teva implementació
  [                                                    ]
}

// Immersió:
/* Pre: p és un punter a l'arrel d'un arbre N-ari no buit */
/* Post: retorna true si i només si tots els subarbres de l'arbre amb
   node arrel apuntat per p compleixen que la suma dels seus valors és
   múltiple del seu nombre de nodes. A més, si la funció retorna true,
   llavors [                                                    ]
 */
static bool ArbreNari::i_mult(node* p, [                    ] ) {
  if ([                    ]) {
    // Cas base
    [                                                    ]
  } else {
     bool compleix_propietat = true;
    // inicialitzacions
    [                                                    ]
    for (int i = 0; i < fill.size() and compleix_propietat; ++i) {
      if (p -> fill[i] != nullptr) {
        [                                                    ]
      }
    }
    if (compleix_propietat)
      // tots els fills compleixen la propietat, cal veure
      // que l'arbre sencer també
      return [                    ] ;
    else
      return false;
  }
}
```