

capses indicades. Cada capsa ha de contenir exactament una instrucció simple o una expressió. S'ha de respectar el següent invariant del bucle:

```
/* Inv: Sigui L la llista implícita, distmax es la distància màxima de dos elements consecutius de
la subllista de L entre primer_node i ant, i primer_distmax apunta al primer d'ells;
antseg = ant->seg */
```

SOLUCIÓ:

```
void Llista::suavitza(){
    node_llista* ant =  ;
    node_llista* antseg =  ;
    int distmax =  ;
    node_llista* primer_distmax =  ;
     ;
     ; // Aquí s'ha de complir l'Inv
    while (  ) {
        int dist =  ;
        if (  ) {
             ;
             ;
        }
         ;
         ;
    }
    node_llista* segon_distmax =  ;
    act =  ;
    act->info =  ;
     ;
     ;
     ;
     ;
    ++longitud;
}
```



```

// public:
//
void Arbre::construeix_ple(int k) {
    // Escriu aquí la teva implementació
    
}

// private:
//
static node_arbre* Arbre::i_construeix_ple(int k, int prof) {
    if (  ) {
        // Cas base
        
    }
    else {
        // Cas recursiu
        node_arbre* p = 
        
        
        
        
    }
}
}

```


Ompliu les capses amb el vostre codi, respectant la resta del codi ja escrit. Cada capsa pot contenir ara més d'una instrucció. El codi no ha de cridar a cap mètode públic de la classe `ArbreGen` i només pot cridar als mètodes privats `i_corregeix` i `esborra_node_arbreGen`.

SOLUCIÓ:

```
// public :  
//  
void ArbreGen::corregeix() {  
    // Escriu aquí la teva implementació  
      
}  
  
// private :  
//  
static void ArbreGen::i_corregeix(node_arbreGen*& p, int prof) {  
    if (  ) {  
        // Cas base  
          
    }  
    else {  
        // Cas recursiu  
          
    }  
}
```