Segundo examen parcial de PRO2 Modelo A Duración: 2h 50m 19/01/2018

Apellidos **Nombre** **DNI**

Problema 1 (6 puntos)

En este problema debéis implementar algunos métodos públicos de la clase *Llista*, cuya implementación habéis visto en clase de teoría. Mostramos a continuación la representación del tipo *Llista*, que utiliza nodos *doblemente encadenaos* con punteros al elemento siguiente (`seg`) y al elemento anterior (`ant`). Esta implementación de la clase *Llista* contiene los siguientes atributos: (1) `longitud`, de tipo entero; (2) `primer_node`, un puntero a `Node` que apunta al nodo que representa el primer elemento de la lista; (3) `ultim_node`, un puntero a `Node` que apunta al nodo que representa el último elemento de la lista; y (4) `act`, un puntero a `Node` que apunta al nodo que representa el elemento actual de la lista, denominado *el punto de interés* de la lista.

```
template <class T> class Llista {
  private:
    struct Node {
        T info;
        Node* seg;
        Node* ant;
    };
    int longitud;
    Node* primer_node;
    Node* ultim_node;
    Node* act;      ...  // especificación e implementación de operaciones privadas
  public:
    ...                  // especificación e implementación de operaciones públicas
};
```

En vuestras respuestas a este problema no podéis utilizar ningún método privado o público de la clase `Llista` que habéis visto en teoría (por ejemplo, `copia_node_llista`, `esborra_node_llista`, `afegir`, `eliminar`, `concat`, `inici`, `fi`, `avanca`, `retrocedeix`, `actual` o `modifica_actual`). Si utilizáis algún método privado o público auxiliar en vuestra respuesta a algún apartado, debéis especificarlo –escribiendo claramente la cabecera, la precondición y la postcondición– e implementarlo en dicha respuesta.

1.1 Definid el método público `push_back(x)`, que añade el elemento x al final de la lista parámetro implícito. Tened en cuenta que el parámetro implícito puede ser una lista vacía o no vacía. Por ejemplo, si x es 6 y a es $\{1,5,3,4,2\}$, después de la llamada `a.push_back(x)`, a debe ser $\{1,5,3,4,2,6\}$. [2 puntos]

```
void push_back(const T& x) {
/* Pre: El parámetro implícito es igual a la lista {e_1,...,e_n}. */
/* Post: El parámetro implícito es igual a la lista {e_1,...,e_n,x}. */
```

1.2 Definid el método público `interseccio_ordenada`, que modifica el parámetro implícito de manera que contenga su intersección con la lista `c2`. Observad que esta operación debe liberar la memoria de todos los nodos que se descarten del parámetro implícito al calcular su intersección con c2. Por ejemplo, si a es la lista $\{1,3,5,7,8,9,10\}$ y b es la lista $\{-1,1,2,3,7,8,9\}$, después de la llamada `a.interseccio_ordenada(b)`, a debe ser la lista $\{1,3,7,8,9\}$ y su punto de interés debe apuntar a 1. [4 puntos]

void *interseccio_ordenada* (**const** *Llista* & *c2*) {
/* Pre: El parámetro implícito es igual a C1. C1 y c2 están ordenadas crecientemente y no contienen elementos repetidos. */
/* Post: El parámetro implícito contiene los elementos de C1 que pertenecen a la intersección de C1 y c2 en el mismo orden en que estaban en C1. El punto de interés del parámetro implícito apunta a su inicio. */

Segundo examen parcial de PRO2 Modelo A Duración: 2h 50m 19/01/2018

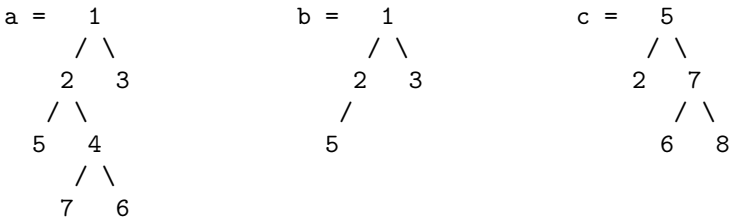
Apellidos **Nombre** **DNI**

Problema 2 (4 puntos)

Implementad eficientemente el método público `poda_subarbre`, especificado a continuación.

```
bool poda_subarbre(int x);
/* Pre: El parámetro implícito es un árbol binario A de enteros. Los valores de los
   nodos de A son todos diferentes. */
/* Post: Si x es el valor de algún nodo de A, el resultado es cierto y el parámetro implícito es
   el resultado de eliminar de A el nodo con valor x y todos sus descendientes; en otro caso,
   el resultado es falso y el parámetro implícito no varía (es decir, es A). */
```

Por ejemplo, si t es igual al árbol a de la figura y x es 4, después de la llamada `t.poda_subarbre(x)`, t debe ser el árbol b de la figura y el resultado cierto. Del mismo modo, si s es el árbol c de la figura y z es 3, después de la llamada `s.poda_subarbre(z)`, s no varía, es decir, s debe ser el árbol c de la figura y el resultado falso.



Damos a continuación la definición del tipo `Arbre`, que debéis utilizar para resolver este problema.

```
template <class T> class Arbre {
  private:
    struct Node_arbre {
      T info;
      Node_arbre* segE;
      Node_arbre* segD;
    };
     Node_arbre* primer_node;
    ...                 // especificación e implementación de operaciones privadas
  public:
    ...                 // especificación e implementación de operaciones públicas
};
```

Si utilizáis algún método privado o público de la clase `Arbre` que habéis visto en clase de teoría (por ejemplo, `copia_node_arbre`, `esborra_node_arbre`, `a_buit`, `es_buit`, `arrel`, `plantar` o `fills`) en vuestra respuesta, debéis especificarlo –escribiendo claramente su cabecera, precondición y postcondición– e implementarlo en dicha respuesta.

Concretamente, se pide implementar eficientemente el método público `poda_subarbre` utilizando varios métodos privados auxiliares que trabajen directamente con datos de tipo `Node_arbre` y de tipo puntero a `Node_arbre`. Debéis

- Escribir la cabecera, la precondición y la postcondición de los métodos auxiliares.
- Implementar los métodos auxiliares.
- Implementar el método público `poda_subarbre` utilizando uno de los métodos auxiliares.

Observad que el método `poda_subarbre` debe liberar la memoria de todos los nodos que se eliminen del parámetro implícito.

bool *poda_subarbre*(**int** *x*);
/* Pre: El parámetro implícito es un árbol binario A de enteros. Los valores de los nodos de A son todos diferentes. */
/* Post: Si x es el valor de algún nodo de A, el resultado es cierto y el parámetro implícito es el resultado de eliminar de A el nodo con valor x y todos sus descendientes; en otro caso, el resultado es falso y el parámetro implícito no varía (es decir, es A). */

Métodos auxiliares: especificación e implementación.