Enunciado:

Resuelva los siguientes ejercicios sobre árboles binarios de búsqueda.

Utilice el estándar C++14 en la solución de sus problemas. No olvide compilar con los *flags* apropiados para detectar *warnings* y errores.

1. Implemente la estructura de datos árbol binario de búsqueda (*binary search tree*) usando recursión. La interface de esta estructura de datos debe tener los siguientes métodos y variables de instancia:

```
template <typename keyType>
class bst {
    struct bstNode {
        keyType key;
        bstNode *left;
        bstNode *right;
        bstNode *parent;
    };

    size_t count;
    bstNode *tree;

    bstNode * min(bstNode *root) const;
    bstNode * max(bstNode *root) const;
    bstNode * successor(bstNode *root) const;
    bstNode * predecessor(bstNode *root) const;

    void remove(bstNode * &root, keyType key);
    bstNode * copy(bstNode *root) const;
    void clear(bstNode * &root);
    void display(bstNode *root, std::ostream &out) const;
    void insert(bstNode * &root, keyType key);
    bstNode * find(bstNode *root, keyType key) const;

public:
    bst();
    bst(const bst &rhs);
    ~bst();

    void remove(keyType key);
    bool empty(void) const;
    void clear(void);
    bool find(keyType key) const;
    void insert(keyType key);
    void display(std::ostream &out = std::cout) const;
};

#include "bst.cpp"
```

Los métodos `min(root)` [`max(root)`] encuentran el menor [mayor] elemento en el árbol de búsqueda binaria cuya raíz es `root`. Este nodo se encuentra siguiendo el camino de punteros definidos por los hijos izquierdos [derechos] que empieza en la raíz y termina en `nullptr`. Los métodos deben retornar un puntero al correspondiente nodo.

El sucesor de un nodo `bush` es definido como el nodo con la menor llave más grande que la llave de `bush`. De forma análoga, el predecesor del nodo `bush` está definido como el nodo con la mayor llave más chica que la llave de `bush`. Si el sucesor (o el predecesor) no existe el método debe retornar `nullptr`.

2. Dado un árbol binario de búsqueda T, escriba los métodos

```
int size() const;              // public
int size(tree_type *T) const; // private
```

público y privado, respectivamente, que calculan el tamaño de T. Es decir, el número total de nodos que contiene T. La implementación no debe hacer uso de atributos que refieran al número de nodos de T.

3. Considere un árbol binario de búsqueda T. Implemente los métodos, público y privado, respectivamente, que calculan la altura de T. Su prototipo debe ser:

```
int height() const;              // public
int height(tree_type *T) const; // private
```

La implementación no debe hacer uso de atributos que refieran al número de nodos de T.

Definición: La altura de un árbol binario T es el número de nodos a lo largo de la trayectoria más larga desde el nodo raíz hasta el nodo hoja más lejano.