

Algoritmos y Estructuras de datos

Ejercicios previos al tercer examen parcial

- Escribe una implementación de la función `altura`, la cual toma un apuntador al nodo raíz de un árbol binario y devuelve la altura de dicho árbol. Las hojas (los nodos sin hijos) apuntan a `nullptr` como sus nodos hijos. Tienes disponibles las funciones de `algorithm` la biblioteca de C++.

```
struct nodo {
    nodo* izq;
    nodo* der;
};

int altura(nodo* p) {
    return (p == nullptr ? 0 : std::max(altura(p->izq), altura(p->der)) + 1);
}
```

- Escribe una implementación de la función `peso`, la cual toma un apuntador al nodo raíz de un árbol binario, y devuelve cuántos nodos tiene el árbol. Las hojas (es decir, los nodos sin hijos) apuntan a `nullptr` como sus nodos hijos.

```
struct nodo {
    nodo* izq;
    nodo* der;
};

int peso(nodo* p) {
    return (p == nullptr ? 0 : cuenta_nodos(p->izq) + cuenta_nodos(p->der) + 1);
}
```

- Escribe una implementación de la función `cuenta_apariciones`, la cual toma un apuntador al nodo raíz de un árbol binario además de un entero y devuelve cuántas veces aparece el entero en el árbol. Las inserciones en el árbol se realizaron como sigue: todos los valores del subárbol izquierdo de un nodo son menores al valor de dicho nodo, mientras que todos los valores del subárbol derecho son mayores o iguales. Las hojas (es decir, los nodos sin hijos) apuntan a `nullptr` como sus nodos hijos. Tu función no debe buscar en partes del árbol donde se sabe (por la regla de inserción) que el entero no aparece.

```
struct nodo {
    int valor;
    nodo* izq;
    nodo* der;
};

int cuenta_apariciones(nodo* p, int v) {
    if (p == nullptr) {
        return 0;
    } else if (v < p->valor) {
        return cuenta_apariciones(p->izq, v);
    } else {
        return cuenta_apariciones(p->der, v) + (p->valor == v);
    }
}
```

- Escribe una implementación de la función `cuenta_hojas`, la cual toma un apuntador al nodo raíz de un árbol binario y devuelve la cantidad de hojas de dicho árbol. Las hojas (es decir, los nodos sin hijos) apuntan a `nullptr` como sus nodos hijos.

```

struct nodo {
    nodo* izq;
    nodo* der;
};

int cuenta_hojas(nodo* p) {
    if (p == nullptr) {
        return 0;
    } else if (p->izq == nullptr && p->der == nullptr) {
        return 1;
    } else {
        return cuenta_hojas(p->izq) + cuenta_hojas(p->der);
    }
}

```

- Escribe una implementación de la función `cuenta_internos`, la cual toma un apuntador al nodo raíz de un árbol binario y devuelve cuántos nodos internos tiene el árbol. Las hojas (es decir, los nodos sin hijos) apuntan a `nullptr` como sus hijos. Un nodo interno es un nodo que no es hoja.

```

struct nodo {
    nodo* izq;
    nodo* der;
};

int cuenta_internos(nodo* p) {
    if (p == nullptr || p->izq == nullptr && p->der == nullptr) {
        return 0;
    } else {
        return cuenta_internos(p->izq) + cuenta_internos(p->der) + 1;
    }
}

```