**PARTE 2: Ejercicios de seudocódigo**

**Duración: 60 minutos**

**CONSIDERACIONES IMPORTANTES**

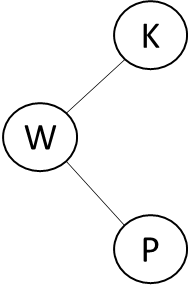
* Se deben respetar las normas publicadas sobre la escritura de seudocódigo (lenguaje natural, pre y post condiciones, seudocódigo detallado sobre esta implementación del TDA.
* Se recomienda escribir métodos auxiliares para simplificar el método principal.
* Se deben desarrollar completamente todas las operaciones invocadas.
* Se deben calcular los órdenes del tiempo de ejecución de los algoritmos desarrollados.

**EJERCICIO 1**

Dado un árbol binario, escribir un algoritmo que devuelva la cantidad de nodos internos (los nodos que no son hoja) no completos (es decir, que les falte un hijo), de acuerdo a la siguiente firma:

TArbolBinario.internosNoCompletos() //devuelve un entero

TNodoArbolBinario.internosNoCompletos() //devuelve un entero

Ejemplo

Si el árbol es:

El algoritmo devuelve: **2** (nodo “K” y nodo “W”)

**EJERCICIO 2**

**ESCENARIO**

Un hospital de nuestro país mantiene una **plantilla** de **médicos** cuyos elementos contienen información relacionada con cada médico que trabaja en la institución: **Cédula**, **nombre**, **especialidad**, **año de ingreso**. La plantilla es almacenada utilizando un Árbol Binario de Búsqueda, que se utiliza habitualmente para emitir listados de todos los médicos **ordenados por número de Cédula**.

Una consulta habitual es solicitar un directorio de todos los médicos de una cierta **especialidad** (ej: cardiología, pediatría, cirugía, etc.), ordenado **alfabéticamente**.

Por ejemplo, si, al iniciar, la **plantilla de médicos** del **Hospital** contiene (por su listado en inorden)

1, (1, “Pedro”, “Cirugía”, 2005);

2, (2, “María”, “Cardiología” , 2004);

3, (3, “José”, “Pediatría” , 2008);

4, (4, “Florencia“, “Pediatría” , 2004);

5, (5, “Juan”, “Cardiología” , 2007);

6, (6, “Ana”, “Cirugía” , 2002);

7, (7, “Martín”, “Pediatría” , 2009);

**luego de ejecutado el algoritmo** que construye el directorio para **“Cardiología”** (en un nuevo ABB), el listado del mismo en inorden es:

Juan, (5, “Juan”, “Cardiología”, 2007);

María, (2, “María”, “Cardiología”, 2004);

**SE PIDE:**

1. Desarrollar un algoritmo del TDA Árbol Binario de Búsqueda (utilizado para una “***plantilla de médicos***”, que extiende el TDA ABB estándar), cuyos elementos son del tipo **TElementoAB** con referencia a objeto del tipo “***Médico***” -con atributos ***cédula, nombre, especialidad y año de ingreso*)**,– y etiqueta correspondiente al campo ***cédula***, que, recibiendo como parámetro un **nombre de especialidad** (ej, “Cardiología”), construya un nuevo ABB que contendrá los médicos de esta especialidad, utilizando como clave de los nodos (clave de inserción / búsqueda) el **nombre** del médico.

**Entidades involucradas**

* **Medico** atributos: Cedula, Nombre, Especialidad, año de ingreso
* **PlantillaMedicos**ABB que agrega el método requerido. Los nodos son del tipo **TDA TElementoAB**(con campo etiqueta = **medico.cedula**, campo Dato referencia un Medico).
* **DirectorioEpecialidad, ABB**de especialidad con los médicos de la especialidad. Los nodos son del tipo **TDA TElementoAB**(con campo etiqueta = **medico.nombre**, campo Dato referencia un Médico)

**Firma de los métodos a desarrollar:**

**De TDA PlantillaMedicos**(que es o desciende de ABB)

**obtenerDirectorio (de tipo string especialidad) : DEVUELVEun TABB**

**CONSIDERACIONES IMPORTANTES**

* Se deben respetar las normas publicadas sobre la escritura de seudocódigo (lenguaje natural, pre y post condiciones, seudocódigo detallado sobre esta implementación del TDA.
* la lista original de médicos no se ve alterada.
* No se deben duplicar instancias de ningún TDA
* Se deben desarrollar completamente todas las operaciones invocadas.

Ejercicio 1

Lenguaje natural :

**De TDA PlantillaMedicos**

**obtenerDirectorio (de tipo string especialidad) : DEVUELVEun TABB**

El método checkeara que la raíz no sea nula, si lo es, devolverá 0. Si no lo es, creara un nuevo TABB vacio y hara una llamada al método “obtenerDirectorio” con el nuevo árbol como parámetro. Como resultado se devolverá este nuevo arbol.

**De TNodoArbolBinario**

**obtenerDirectorio (de tipo TABB nuevoArbol, de tipo String especialidad)**

El método recorrerá el árbol existente con todos los médicos y en cada caso revisara si su especialidad es la buscada en caso de no serlo seguirá al próximo caso, pero en caso de serlo, creara un nuevo nodo en el que la etiqueta será el nombre del medico y agregara este nuevo nodo al árbol que fue pasado como parámetro.

Precondiciones :

**De TDA PlantillaMedicos**

**obtenerDirectorio (de tipo string especialidad) : DEVUELVEun TABB**

* Que el árbol de médicos no sea vacio
* Que todos los médicos tengas una especialidad

**De TNodoArbolBinario**

**obtenerDirectorio (de tipo TABB nuevoArbol, de tipo String especialidad)**

* Que los datos guardados sean medicos

PostCondiciones :

**De TDA PlantillaMedicos**

**obtenerDirectorio (de tipo string especialidad) : DEVUELVEun TABB**

* El resultado sera un árbol con todos los médicos que existan de una misma especialidad
* Los médicos encontrados estarán ordenados por nombre

**De TNodoArbolBinario**

**obtenerDirectorio (de tipo TABB nuevoArbol, de tipo String especialidad)**

* La estructura del árbol no será modificada luego de recorrida

Seudocodigo :

**De TDA PlantillaMedicos**

**obtenerDirectorio (de tipo string especialidad) : DEVUELVEun TABB**

Si (raíz <> nulo) Hacer

nuevoArbol de tipo TABB nuevoArbol

raíz.obtenerDirectorio(nuevoArbol, especialidad)

devolver nuevoArbol

Sino Hacer

Devolver null

Fin Si

Fin

**De TNodoArbolBinario**

**obtenerDirectorio (de tipo TABB nuevoArbol, de tipo String especialidad)**

Si (dato.especialidad es igual a especialidad) Hacer

De tipo TNodoArbolBinario nuevoNodo(

Etiqueta = medico.nombre,

Dato = medico

)

nuevoArbol.agregar(nuevoNodo)

Fin Si

Si (hijoDer <> nulo) Hacer

hijoDer.obtenerDirectorio(nuevoArbol, especialidad)

Fin Si

Si (hijoIz <> nulo) Hacer

hijoIz.obtenerDirectorio(nuevoArbol, especialidad)

Fin Si

Fin

Calculo de Orden :

**De TDA PlantillaMedicos**

**obtenerDirectorio (de tipo string especialidad) : DEVUELVEun TABB**

Es O(n) ya que en el peor de los casos se ejecutara el método obtenerDirectorio que tiene orden O(n).

**De TNodoArbolBinario**

**obtenerDirectorio (de tipo TABB nuevoArbol, de tipo String especialidad)**

Este método es de orden O(n) siendo n la cantidad de nodos del árbol. Esto es así ya que el método visitará todos los nodos del árbol.

Ejercicio 1

Lenguaje natural :

TArbolBinario.internosNoCompletos()

El método checkeara que la raíz no sea nula, si lo es, devolverá 0. Si no lo es, como resultado devolverá lo que la llamada al método “internosNoCompletos” del nodo raíz devuelva.

TNodoArbolBinario.internosNoCompletos()

El método funcionara siguiendo una lógica recursiva. Primero se buscara el caso base, que en este caso es que el nodo sea una hoja, en este caso el método devolverá 0. Luego se checkeara si el nodo en el que me eencuentro es un nodo incompleto (existen dos posibles casos, solo hijo izquierdo o solo hijo derecho) en cualquiera de los dos casos el método devolverá lo que la llamada a su nodo hijo existente devuelva y además se le sumara una unidad. Luego, en el caso de que tenga dos hijos, simplemente se devolverá la suma de lo que devuelvan las dos llamadas de este mismo método de los nodos hijos.

Precondiciones :

TArbolBinario.internosNoCompletos()



TNodoArbolBinario.internosNoCompletos()

PostCondiciones :

TArbolBinario.internosNoCompletos()

* El método devolverá el numero de nodos incompletos del árbol y 0 en caso de vacio

TNodoArbolBinario.internosNoCompletos()

* El método devolverá 0 en caso de nodo hoja o un entero que representara la cantidad de nodos hijos incompletos de ese nodo
* El árbol no cambiara su estructura luego de ejecutado

Seudocodigo :

TArbolBinario.internosNoCompletos() : devuelve un entero

Comienza

Si (raíz <> nulo) Hacer

Devovler raíz.internosNoCompletos()

Sino Hacer

Devolver 0

Fin Si

Fin

TNodoArbolBinario.internosNoCompletos(): devuelve un entero

Comienza

Si (hijoIz es nulo y hijoDer es nulo) Hacer

Devolver 0

Sino Si (hijoIz <> nulo y hijoDer <> nulo) Hacer

Devolver hijoIz.internosNoCompletos() + hijoDer.internosNoCompletos()

Sino Si (hijoIz es nulo y hijoDer <> nulo) Hacer

Devolver 1 + hijoDer.internosNoCompletos()

Sino

Devolver hijoIz.internosNoCompletos() + 1

Fin Si

Fin

Calculo de Orden :

TArbolBinario.internosNoCompletos() : devuelve un entero

Es O(n) ya que en el peor de los casos se ejecutara el método internosNoCompletos que tiene ornden O(n).

TNodoArbolBinario.internosNoCompletos(): devuelve un entero

Este método es de orden O(n) siendo n la cantidad de nodos del árbol. Esto es así ya que el método visitará todos los nodos del árbol en busca de los nodos que sean incompletos.