EJERCICIO 1

Int getNodos(Abin<T>A)

Return getNodos\_rec(A.raiz(), A)

Int getNodos\_rec(Abin<T>)

If(n == Abin<T>::NODO\_NULO)

Return 0;

Else

// 1+IZQ+DER

1+getNodos\_rec(A.hijoIzqdo(A), A) + getNodos\_rec(A.hijoDrch(A), A)

// si me encuentro un nodo nulo devuelvo 0 si no es nulo sumo uno

Otra definición Altura: máxima altura izquierda y derecha + 1

EJERCICIO 2

If(n == Abin<T>::NODO\_NULO)

Return -1 // Para casos conflictivos como cuando solo tenemos un nodo en el árbol

Else

1+max(alturaSubarbol(), alturaSubarbol())

Si delvolvieramos 0 cuando solo hay un nodo la función nos devolvería 1+max(…) = 1+ 0 = 1, esto sería erroneo ya que en este caso nos tendría que devolver 0, al devolver -1 se le resta 1 al resultado y ya estaría bien

EJERCICIO 3

Int getDeep(Abin<T>::nodo n, Abin<T>A)

If(n == A.raiz())

Return 0;

Else

Return 1+ getDeep( A.padre(n), A)

EJERCICIO 4 (Ahora a nivel de desarrollador del TAD)

A.padre(aux) no tendría sentido hacerlo porque es más costoso que acceder directamente a la variable privada del TAD ya que ahora podemos

EJERCICIO 5

EJERCICIO 6

De nuevo nos piden una función que reciba un árbol, por lo que tendremos que hacer 2, ya que la solución es recursiva.

Si el árbol esta vacio lo controlamos en la función no recursiva(primera función)

Max (desequilbrio(), desequilibrioDrch(), diferenciaAlturas) No hay max de 3 elementos por lo que anidamos max

EJERCICIO 7

Nos pide una función que recibe un árbol por lo que tenemos que hacer dos funciones, la llamada y la función recursiva que la resuelve

ÁRBOL PSEUDOCOMPLETO El penúltimo nivel del árbol tiene dos hijos o ninguno

If NODO\_NULO // Árbol vacío

If Altura == 0 // Nodo raíz

If Altura == 1 && (nHijos == 0)

Return true