

Apellidos:
Nombre:
Convocatoria:
DNI:

Examen PED enero 2006

Modalidad 0

- Normas:**
- La entrega del test **no** corre convocatoria.
 - Tiempo para efectuar el test: **35 minutos**.
 - Una pregunta mal contestada elimina una correcta.
 - Las soluciones al examen se dejarán en el campus virtual.
 - Una vez empezado el examen no se puede salir del aula hasta finalizarlo.**
 - En la **hoja de contestaciones** el verdadero se corresponderá con la **A**, y el falso con la **B**.

	V	F	
En la especificación de un TAD, las operaciones auxiliares son visibles para los usuarios.	<input type="checkbox"/>	<input type="checkbox"/>	1. F
Una operación del TAD X que tenga la sintaxis Crear() →X es una operación constructora generadora.	<input type="checkbox"/>	<input type="checkbox"/>	2. V
En C++, los miembros <i>protected</i> son privados para el exterior, pero permiten el acceso a las clases derivadas.	<input type="checkbox"/>	<input type="checkbox"/>	3. V
Dadas las clases <i>TDir</i> y <i>TVectorDir</i> :	<input type="checkbox"/>	<input type="checkbox"/>	4. F
<pre>class TDir { public: private: int e1; char c1; };</pre>	<input type="checkbox"/>	<input type="checkbox"/>	5. F
<pre>class TVectorDir { public: private: TDir *vector; int dim; };</pre>	<input type="checkbox"/>	<input type="checkbox"/>	6. V
<pre>TVectorDir::~TVectorDir () { if (v!=NULL) delete v; dim=0; v=NULL; }</pre>	<input type="checkbox"/>	<input type="checkbox"/>	7. F
¿Es correcta la implementación del destructor de <i>TVectorDir</i> ?	<input type="checkbox"/>	<input type="checkbox"/>	8. F
El nivel de un nodo en un árbol coincide con la longitud del camino desde la raíz a dicho nodo	<input type="checkbox"/>	<input type="checkbox"/>	9. V
Dado un único recorrido de un árbol binario lleno es posible reconstruir dicho árbol	<input type="checkbox"/>	<input type="checkbox"/>	10. V
Sea el tipo <i>arbin</i> definido en clase. La semántica de la operación <i>nodos</i> es la siguiente: Var i,d:arbin; x:item; nodos(crear_arbin())=0 nodos(enraizar(i,x,d))=nodos(i)+nodos(d)	<input type="checkbox"/>	<input type="checkbox"/>	11. F
A los árboles generales también se les llama árboles multicamino de búsqueda	<input type="checkbox"/>	<input type="checkbox"/>	12. V
El ítem medio (según la relación de orden) almacenado en un árbol binario de búsqueda lleno siempre se encuentra en la raíz.	<input type="checkbox"/>	<input type="checkbox"/>	13. V
Un árbol completo siempre está balanceado respecto a la altura	<input type="checkbox"/>	<input type="checkbox"/>	14. V
En un árbol AVL siempre que se inserte una etiqueta hay que realizar una rotación	<input type="checkbox"/>	<input type="checkbox"/>	15. F
El coste temporal en el peor caso de la operación de inserción en un árbol 2-3-4 es $\log_2(n+1) \approx \log_2(n)$ siendo “n” el número total de ítems	<input type="checkbox"/>	<input type="checkbox"/>	16. V
Se puede obtener un único árbol 2-3-4 a partir de su recorrido por niveles	<input type="checkbox"/>	<input type="checkbox"/>	17. F
La semántica de la operación <i>anterior</i> vista en clase es la siguiente: VAR L1: lista; x: ítem; p: posición; anterior(L1, primera(L1)) = error_posicion(); si p != ultima(L1) entonces anterior(L1, siguiente(L1, p)) = p anterior(inscabeza(L1, x), primera(L1)) = primera(inscabeza(L1, x))	<input type="checkbox"/>	<input type="checkbox"/>	18. F
Sea el tipo <i>vector</i> definido en clase. La semántica de la operación <i>recu</i> es la siguiente: Var v:vector; i,j:int; x:ítem; recu(crear_vector(),i)=error_item(); si i<>j entonces recu(asig(v,i,x),j)=recu(v,j) sino recu(asig(v,i,x),j)=TRUE	<input type="checkbox"/>	<input type="checkbox"/>	
<i>crear_pila()</i> , <i>apilar(pila,item)</i> y <i>desapilar(pila)</i> son operaciones constructoras del tipo <i>pila</i> .	<input type="checkbox"/>	<input type="checkbox"/>	
Todo árbol binario de búsqueda es un árbol 2-3-4	<input type="checkbox"/>	<input type="checkbox"/>	
La semántica de la operación <i>Base</i> que actúa sobre una <i>pila</i> y devuelve el primer elemento apilado es la siguiente: Base(crear_pila ()) = crear_pila () Base(apilar(crear_pila (), x)) = x Base(apilar(p, x)) = Base(p)	<input type="checkbox"/>	<input type="checkbox"/>	

Apellidos:
Nombre:
Convocatoria:
DNI:

Examen PED febrero 2007

Modalidad 0

- Normas:**
- La entrega del test no corre convocatoria.
 - Tiempo para efectuar el test: **35 minutos**.
 - Una pregunta mal contestada elimina una correcta.
 - Las soluciones al examen se dejarán en el campus virtual.
 - Una vez empezado el examen no se puede salir del aula hasta finalizarlo.**
 - En la **hoja de contestaciones** el verdadero se corresponderá con la **A**, y el falso con la **B**.

V	F		
<input type="checkbox"/>	<input type="checkbox"/>	1	V
<input type="checkbox"/>	<input type="checkbox"/>	2	F
<input type="checkbox"/>	<input type="checkbox"/>	3	F
<input type="checkbox"/>	<input type="checkbox"/>	4	V
<input type="checkbox"/>	<input type="checkbox"/>	5	F
<input type="checkbox"/>	<input type="checkbox"/>	6	V
<input type="checkbox"/>	<input type="checkbox"/>	7	V
<input type="checkbox"/>	<input type="checkbox"/>	8	F
<input type="checkbox"/>	<input type="checkbox"/>	9	F
<input type="checkbox"/>	<input type="checkbox"/>	10	V
<input type="checkbox"/>	<input type="checkbox"/>	11	V
<input type="checkbox"/>	<input type="checkbox"/>	12	F
<input type="checkbox"/>	<input type="checkbox"/>	13	F
<input type="checkbox"/>	<input type="checkbox"/>	14	V

En C++ y cuando se emplea composición (*layering*), los métodos de la clase derivada pueden acceder a la parte pública de la clase base.

En C++, el puntero *this* no se puede emplear para modificar el objeto al que apunta.

En C++, los constructores se pueden invocar explícitamente cuando el programador lo desee (por ejemplo: *TLista a; a.TLista();*).

En C++, la siguiente declaración es incorrecta: *int& a = I;*

En la escala de complejidades se cumple que $O(\log n) \subset O(\log \log n)$.

El algoritmo de búsqueda binaria estudiado en clase (búsqueda de un elemento en un vector ordenado) tiene una complejidad de $\Omega(1)$.

Para un vector de naturales, se define la operación *eliminar* que borra las posiciones pares del vector marcándolas con "0" (para calcular el resto de una división, se puede utilizar la operación MOD). La sintaxis y la semántica de la operación *eliminar* es la siguiente:

```

eliminar: vector → vector
Var v:vector; i: entero; x:natural;
eliminar(crear_vector()) = crear_vector()
si (i MOD 2) == 0
entonces eliminar(asignar(v,i,x)) = asignar(eliminar(v),i,0)
si no eliminar(asignar(v,i,x)) = asignar(eliminar(v),i,x)

```

La semántica de la operación *ultima* vista en clase es la siguiente:

```

VAR L1: lista; x: item;
si esvacia( L1 ) entonces
ultima( inscabeza( L1, x ) = primera( L1 )
si no ultima( inscabeza( L1, x ) = ultima( L1 )

```

Es posible reconstruir un único árbol binario de altura 6 a partir de un recorrido en preorden con 62 etiquetas.

La sintaxis y semántica de la operación *quita_hojas*, que actúa sobre un árbol binario y devuelve el árbol binario original sin sus hojas, es la siguiente:

```

quita_hojas(arbin) → arbin
VAR i, d: arbin; x: item;
quita_hojas(crea_arbin( )) = crea_arbin( )
quita_hojas(enraizar(crea_arbin( ), x, crea_arbin( )) = crea_arbin( )
quita_hojas(enraizar(i, x, d)) =
enraizar(quita_hojas(i), x, quita_hojas(d))

```

Dados los recorridos de preorden, postorden y niveles de un árbol binario de altura 2 y 1 hoja es posible reconstruir 2 árboles binarios.

Todo árbol AVL es un árbol 2-3-4

La operación (*DIVIDEHIJODE2 (p, q)*) en la inserción de un elemento en un árbol 2-3-4 puede aumentar la altura del árbol original.

En el algoritmo del borrado de un elemento en un árbol 2-3-4 si *q* es 2-nodo y *r* es 3-nodo hay que hacer una ROTACIÓN.

Apellidos:

Nombre:

Convocatoria:

DNI:

Examen PED febrero 2008

Modalidad 0

- Normas:**
- La entrega del test no corre convocatoria.
 - Tiempo para efectuar el test: **30 minutos**.
 - Una pregunta mal contestada elimina una correcta.
 - Las soluciones al examen se dejarán en el campus virtual.
 - **Una vez empezado el examen no se puede salir del aula hasta finalizarlo.**
 - En la **hoja de contestaciones** el verdadero se corresponderá con la **A**, y el falso con la **B**.

	V	F		
La complejidad logarítmica aparece en algoritmos que descartan muchos valores (generalmente la mitad) en un único paso.	<input type="checkbox"/>	<input type="checkbox"/>	1	V
Dada la sintaxis de la función $IC(lista,item) \rightarrow lista$, que inserta un elemento a la cabeza de la lista pasada como parámetro y $crear() \rightarrow lista$, que crea una lista vacía. La siguiente secuencia: $IC(IC(crear()),a),b,c$, daría como resultado una lista con los elementos en este orden: $a \rightarrow b \rightarrow c$, donde a es el primer elemento de la lista	<input type="checkbox"/>	<input type="checkbox"/>	2	F
Dentro de la especificación algebraica de los números naturales definimos la sintaxis de la función F como: $F: natural \rightarrow BOOL$, y su semántica como: $F(cero)=TRUE$, $F(suc(cero))=FALSE$, $F(suc(suc(x)))=F(x)$. Para el número natural $x=35$, la función F devolvería TRUE.	<input type="checkbox"/>	<input type="checkbox"/>	3	F
En C++, si un objeto se sale de ámbito entonces se invoca automáticamente al destructor de ese objeto.	<input type="checkbox"/>	<input type="checkbox"/>	4	V
En C++, el constructor de copia recibe como argumento un objeto del mismo tipo pasado por referencia o por valor.	<input type="checkbox"/>	<input type="checkbox"/>	5	F
El algoritmo de búsqueda binaria estudiado en clase (búsqueda de un elemento en un vector ordenado) tiene una complejidad de $\Omega(1)$.	<input type="checkbox"/>	<input type="checkbox"/>	6	V
La complejidad espacial es la cantidad de recursos espaciales que un algoritmo consume o necesita para su ejecución	<input type="checkbox"/>	<input type="checkbox"/>	7	V
La operación BorrarItem, que borra todas las ocurrencias del item i que se encuentren en la lista, tiene la siguiente sintaxis y semántica: BorrarItem: LISTA, ITEM -> LISTA BorrarItem(Crear, i) = Crear BorrarItem(IC(L1,j), i) = si (i == j) entonces BorrarItem (L1, i) sino IC (BorrarItem (L1, i), j)	<input type="checkbox"/>	<input type="checkbox"/>	8	V
La semántica de la operación obtener en una lista con acceso por posición es la siguiente (IC= InsertarCabeza(Lista, Ítem), p: posición, l1: lista, x: ítem): obtener(crear(),p)=error_item() si p == primera(IC(l1,x)) entonces obtener(IC(l1,x),p)=x sino obtener(IC(l1,x),p)=IC(obtener(l1,p),x)	<input type="checkbox"/>	<input type="checkbox"/>	9	F
El máximo número de nodos en un nivel $i-1$ de un árbol binario es 2^{i-2} , $i \geq 2$	<input type="checkbox"/>	<input type="checkbox"/>	10	V
Sea el TIPO arbin definido en clase. La semántica de la operación nodos es la siguiente: Var i,d:arbin; x:item; nodos(crear_arbin())=0 nodos(enraizar(i,x,d))=nodos(i)+nodos(d)	<input type="checkbox"/>	<input type="checkbox"/>	11	F
El coste temporal en su peor caso de insertar una etiqueta en un árbol binario de búsqueda es lineal con la altura del árbol	<input type="checkbox"/>	<input type="checkbox"/>	12	V
Se puede obtener un único árbol 2-3-4 a partir de su recorrido por niveles	<input type="checkbox"/>	<input type="checkbox"/>	13	V

Apellidos:
Nombre:
Convocatoria:
DNI:

Examen PED febrero 2008

Modalidad 1

- Normas:**
- La entrega del test **no** corre convocatoria.
 - * Tiempo para efectuar el test: **30 minutos**.
 - * Una pregunta mal contestada elimina una correcta.
 - * Las soluciones al examen se dejarán en el campus virtual.
 - * **Una vez empezado el examen no se puede salir del aula hasta finalizarlo.**
 - * En la **hoja de contestaciones** el verdadero se corresponderá con la **A**, y el falso con la **B**.

V	F		
<input type="checkbox"/>	<input type="checkbox"/>	1	V
<input type="checkbox"/>	<input type="checkbox"/>	2	V
<input type="checkbox"/>	<input type="checkbox"/>	3	F
<input type="checkbox"/>	<input type="checkbox"/>	4	F
<input type="checkbox"/>	<input type="checkbox"/>	5	V
<input type="checkbox"/>	<input type="checkbox"/>	6	F
<input type="checkbox"/>	<input type="checkbox"/>	7	V
<input type="checkbox"/>	<input type="checkbox"/>	8	V
<input type="checkbox"/>	<input type="checkbox"/>	9	F
<input type="checkbox"/>	<input type="checkbox"/>	10	V
<input type="checkbox"/>	<input type="checkbox"/>	11	V
<input type="checkbox"/>	<input type="checkbox"/>	12	V
<input type="checkbox"/>	<input type="checkbox"/>	13	F

El algoritmo de búsqueda binaria estudiado en clase (búsqueda de un elemento en un vector ordenado) tiene una complejidad de $\Omega(1)$.
 El coste temporal en su peor caso de insertar una etiqueta en un árbol binario de búsqueda es lineal con la altura del árbol
 Dada la sintaxis de la función $IC(lista,item) \rightarrow lista$, que inserta un elemento a la cabeza de la lista pasada como parámetro y $crear() \rightarrow lista$, que crea una lista vacía. La siguiente secuencia: $IC(IC(IC(crear()),a),b),c$, daría como resultado una lista con los elementos en este orden: $a \rightarrow b \rightarrow c$, donde a es el primer elemento de la lista
 Dentro de la especificación algebraica de los números naturales definimos la sintaxis de la función F como: $F: natural \rightarrow BOOL$, y su semántica como: $F(cero)=TRUE$, $F(suc(cero))=FALSE$, $F(suc(suc(x)))=F(x)$. Para el número natural $x=35$, la función F devolvería $TRUE$.
 El máximo número de nodos en un nivel $i-1$ de un árbol binario es 2^{i-2} , $i \geq 2$
 En C++, el constructor de copia recibe como argumento un objeto del mismo tipo pasado por referencia o por valor.
 En C++, si un objeto se sale de ámbito entonces se invoca automáticamente al destructor de ese objeto.
 La operación **BorrarItem**, que borra todas las ocurrencias del item i que se encuentren en la lista, tiene la siguiente sintaxis y semántica:
 BorrarItem: LISTA, ITEM \rightarrow LISTA
 BorrarItem(Crear, i) = Crear
 BorrarItem(IC(L1,j), i) = si ($i == j$) entonces BorrarItem (L1, i)
 sino IC (BorrarItem (L1, i), j)
 La semántica de la operación obtener en una lista con acceso por posición es la siguiente (IC=InsertarCabeza(Lista, Ítem), p: posición, l1: lista, x: ítem):
 obtener(crear(),p)=error_item()
 si $p == \text{primera}(IC(l1,x))$ entonces obtener(IC(l1,x),p)=x
 sino obtener(IC(l1,x),p)=IC(obtener(l1,p),x)
 La complejidad espacial es la cantidad de recursos espaciales que un algoritmo consume o necesita para su ejecución
 La complejidad logarítmica aparece en algoritmos que descartan muchos valores (generalmente la mitad) en un único paso.
 Se puede obtener un único árbol 2-3-4 a partir de su recorrido por niveles
 Sea el TIPO arbin definido en clase. La semántica de la operación nodos es la siguiente:
 Var i,d:arbin; x:item;
 nodos(crear_arbin())=0
 nodos(enraizar(i,x,d))=nodos(i)+nodos(d)

Apellidos:

Nombre:

Convocatoria:

DNI:

Examen PED febrero 2008

Modalidad 2

Normas: • La entrega del test no corre convocatoria.

* Tiempo para efectuar el test: **30 minutos**.

* Una pregunta mal contestada elimina una correcta.

* Las soluciones al examen se dejarán en el campus virtual.

* **Una vez empezado el examen no se puede salir del aula hasta finalizarlo.**

* En la **hoja de contestaciones** el verdadero se corresponderá con la **A**, y el falso con la **B**.

	V	F	
Dentro de la especificación algebraica de los números naturales definimos la sintaxis de la función F como: F: natural \rightarrow BOOL, y su semántica como: F(cero)=TRUE, F(suc(cero))=FALSE, F(suc(suc(x)))=F(x). Para el número natural x=35, la función F devolvería TRUE.	<input type="checkbox"/>	<input type="checkbox"/>	1 F
La operación BorrarItem, que borra todas las ocurrencias del item i que se encuentren en la lista, tiene la siguiente sintaxis y semántica: BorrarItem: LISTA, ITEM -> LISTA BorrarItem(Crear, i) = Crear BorrarItem(IC(L1,j), i) = si (i == j) entonces BorrarItem (L1, i) sino IC (BorrarItem (L1, i), j)	<input type="checkbox"/>	<input type="checkbox"/>	2 V
La complejidad logarítmica aparece en algoritmos que descartan muchos valores (generalmente la mitad) en un único paso.	<input type="checkbox"/>	<input type="checkbox"/>	3 V
La complejidad espacial es la cantidad de recursos espaciales que un algoritmo consume o necesita para su ejecución	<input type="checkbox"/>	<input type="checkbox"/>	4 V
En C++, si un objeto se sale de ámbito entonces se invoca automáticamente al destructor de ese objeto.	<input type="checkbox"/>	<input type="checkbox"/>	5 V
En C++, el constructor de copia recibe como argumento un objeto del mismo tipo pasado por referencia o por valor.	<input type="checkbox"/>	<input type="checkbox"/>	6 F
El máximo número de nodos en un nivel $i-1$ de un árbol binario es 2^{i-2} , $i \geq 2$	<input type="checkbox"/>	<input type="checkbox"/>	7 V
Sea el TIPO arbin definido en clase. La semántica de la operación nodos es la siguiente: Var i,d:arbin; x:item; nodos(crear_arbin())=0 nodos(enraizar(i,x,d))=nodos(i)+nodos(d)	<input type="checkbox"/>	<input type="checkbox"/>	8 F
Se puede obtener un único árbol 2-3-4 a partir de su recorrido por niveles	<input type="checkbox"/>	<input type="checkbox"/>	9 V
El coste temporal en su peor caso de insertar una etiqueta en un árbol binario de búsqueda es lineal con la altura del árbol	<input type="checkbox"/>	<input type="checkbox"/>	10 V
La semántica de la operación obtener en una lista con acceso por posición es la siguiente (IC=InsertarCabeza(Lista, Ítem), p: posición, l1: lista, x: ítem): obtener(crear(),p)=error_item() si p == primera(IC(l1,x)) entonces obtener(IC(l1,x),p)=x sino obtener(IC(l1,x),p)=IC(obtener(l1,p),x)	<input type="checkbox"/>	<input type="checkbox"/>	11 F
El algoritmo de búsqueda binaria estudiado en clase (búsqueda de un elemento en un vector ordenado) tiene una complejidad de $\Omega(1)$.	<input type="checkbox"/>	<input type="checkbox"/>	12 V
Dada la sintaxis de la función $IC(lista,item) \rightarrow lista$, que inserta un elemento a la cabeza de la lista pasada como parámetro y $crear() \rightarrow lista$, que crea una lista vacía. La siguiente secuencia: $IC(IC(IC(crear(),a),b),c)$, daría como resultado una lista con los elementos en este orden: $a \rightarrow b \rightarrow c$, donde a es el primer elemento de la lista	<input type="checkbox"/>	<input type="checkbox"/>	13 F

Apellidos:

Nombre:

Convocatoria:

DNI:

Examen PED febrero 2009

Modalidad 0

- Normas:**
- La entrega del test no corre convocatoria.
 - Tiempo para efectuar el test: **25 minutos**.
 - Una pregunta mal contestada elimina una correcta.
 - Las soluciones al examen se dejarán en el campus virtual.
 - Una vez empezado el examen no se puede salir del aula hasta finalizarlo.**
 - En la **hoja de contestaciones** el verdadero se corresponderá con la **A**, y el falso con la **B**.

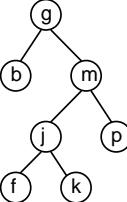
	V	F		
En C++, al declarar una clase "A" como AMIGA de otra clase "B" , todas las funciones miembro de "B" automáticamente pasan a ser funciones AMIGAS de "A"	<input type="checkbox"/>	<input type="checkbox"/>	1	F
En C++, si una clase "B" se construye por composición (layering) , a partir de otra clase "A" , definiendo un objeto miembro de la clase "A" en su parte privada, al invocar al constructor de "B" se invoca antes al constructor de "A" y luego al de "B"	<input type="checkbox"/>	<input type="checkbox"/>	2	V
Las funciones y clases amigas se tienen que declarar en la parte pública de la clase.	<input type="checkbox"/>	<input type="checkbox"/>	3	F
Para el siguiente algoritmo, la complejidad temporal en su peor caso sería O(1):	<input type="checkbox"/>	<input type="checkbox"/>	4	V
<pre>for (i=0; i<100; i++) for (j=0; j<100; j++) if (v[i]<v[j]) v[i]=v[j]; else v[j]=v[i];</pre>	<input type="checkbox"/>	<input type="checkbox"/>	5	V
La complejidad temporal en su peor caso del siguiente fragmento de código es O(n)	<input type="checkbox"/>	<input type="checkbox"/>	6	V
<pre>int i, j, n, sum; for (i = 4; i < n; i++) { for (j = i-3; sum = a[i-4]; j <= i; j++) sum += a[j]; cout << "La suma del subarray " << i-4 << " es " << sum << endl; }</pre>	<input type="checkbox"/>	<input type="checkbox"/>	7	F
La semántica de la operación esvacíapos del tipo vector vista en clase es la siguiente:	<input type="checkbox"/>	<input type="checkbox"/>	8	F
<pre>VAR v: vector; i, j: int; x: item; esvacíapos(crear(), i) = CIERTO esvacíapos(asig(v, i, x), j) si (i == j) entonces FALSO si no esvacíapos(v, j) fsi</pre>	<input type="checkbox"/>	<input type="checkbox"/>	9	F
La semántica de la operación anterior vista en clase es la siguiente:	<input type="checkbox"/>	<input type="checkbox"/>		
<pre>VAR L1: lista; x: item; p: posicion; anterior(L1, primera(L1)) = error_posicion(); si p != ultima(L1) entonces anterior(L1, siguiente(L1, p)) = p anterior(inscabeza(L1, x), primera(L1)) = L1</pre>	<input type="checkbox"/>	<input type="checkbox"/>		
La sintaxis y semántica de la operación simétricos, que comprueba que 2 árboles binarios son simétricos, es la siguiente:	<input type="checkbox"/>	<input type="checkbox"/>		
<pre>simétricos(arbin, arbin) → bool VAR i1, d1, i2, d2: arbin; x, y: item; simétricos(enraizar(i1, x, d1), crea_arbin()) = FALSO simétricos(crea_arbin(), enraizar(i1, x, d1)) = FALSO simétricos(enraizar(i1, x, d1), enraizar(i2, y, d2)) = si (x == y) entonces (simétricos(i1, d2) & simétricos (d1, i2)) sino FALSO</pre>	<input type="checkbox"/>	<input type="checkbox"/>		
Un árbol binario completo es un AVL	<input type="checkbox"/>	<input type="checkbox"/>		

Apellidos, Nombre:
DNI:

Examen PED enero 2010

Modalidad 0

- Normas:**
- La entrega del test **no** corre convocatoria.
 - Tiempo para efectuar el test: **15 minutos**.
 - Una pregunta mal contestada elimina una correcta.
 - Las soluciones al examen se dejarán en el campus virtual.
 - **Una vez empezado el examen no se puede salir del aula hasta finalizarlo.**
 - En la **hoja de contestaciones** el verdadero se corresponderá con la **A**, y el falso con la **B**.

V	F	
<input type="checkbox"/>	<input type="checkbox"/>	1 V
En C++, el valor de la variable q al finalizar este fragmento de código es 11:		
<pre>int q = 0; int k = 5; do { q += k; k++; } while(q < 7);</pre>		
La complejidad temporal (en su caso mejor) del siguiente fragmento de código es $\Omega(n)$		2 V
<pre>int i, length, n, i1, i2, k; for (i = 0, length = 1; i < n-1; i++) { for (i1 = i2 = k = i; k < n-1 && a[k] < a[k+1]; k++, i2++); if (length < i2 - i1 + 1) length = i2 - i1 + 1; }</pre>		
La semántica de la operación insertar del tipo lista vista en clase es la siguiente:		3 F
<pre>VAR L1: lista; x,y: item; p: posicion; insertar(crear(), p, x) = crear() si p == primera(inscabeza(L1, x)) entonces insertar(inscabeza(L1, x), p, y) = inscabeza(inscabeza(L1, x), y) si no insertar(inscabeza(L1, x), p, y) = inscabeza(insertar(L1, p, y), x)</pre>		
El grado de un árbol es el grado mínimo de todos los nodos de ese árbol		4 F
El siguiente árbol es binario de búsqueda		5 F
 <pre>graph TD g((g)) --- b((b)) g --- m((m)) b --- f((f)) b --- j((j)) m --- p((p)) j --- k((k))</pre>		
Dada la siguiente representación secuencial del árbol binario A, [14 8 19 5] el elemento 5 es el hijo izquierda del elemento 8		6 V
En el algoritmo de borrado de un elemento de un árbol AVL, tenemos que actualizar los factores de equilibrio de todos los nodos que han intervenido en la búsqueda del elemento a borrar		7 V
En el algoritmo del borrado de un elemento en un árbol 2-3-4 siempre que q sea 2-nodo hay que hacer una reestructuración.		8 V
El árbol 2-3-4 no vacío tiene como mínimo dos claves en cada nodo		9 F
La operación <i>BorrarItem</i> , que borra todas las ocurrencias del item <i>i</i> que se encuentren en la lista, tiene la siguiente sintaxis y semántica:		10 V
<i>BorrarItem</i> : LISTA, ITEM -> LISTA <i>BorrarItem</i> (Crear, i) = Crear <i>BorrarItem</i> (IC(L1,j), i) = si (i == j) entonces <i>BorrarItem</i> (L1, i) sino IC (<i>BorrarItem</i> (L1, i), j)		

Apellidos, Nombre:
DNI:

Examen PED abril 2013

Modalidad 0

Normas:

- Tiempo para efectuar el test: **25 minutos**.
- Una pregunta mal contestada elimina una correcta.
- Las soluciones al examen se dejarán en el campus virtual.
- **Una vez empezado el examen no se puede salir del aula hasta finalizarlo.**
- En la hoja de contestaciones el verdadero se corresponderá con la **A**, y el falso con la **B**.

V	F		
<input type="checkbox"/>	<input type="checkbox"/>	1	V
<input type="checkbox"/>	<input type="checkbox"/>	2	F
<input type="checkbox"/>	<input type="checkbox"/>	3	V
<input type="checkbox"/>	<input type="checkbox"/>	4	F
<input type="checkbox"/>	<input type="checkbox"/>	5	F
<input type="checkbox"/>	<input type="checkbox"/>	6	V
<input type="checkbox"/>	<input type="checkbox"/>	7	V
<input type="checkbox"/>	<input type="checkbox"/>	8	F
<input type="checkbox"/>	<input type="checkbox"/>	9	F
<input type="checkbox"/>	<input type="checkbox"/>	10	F
<input type="checkbox"/>	<input type="checkbox"/>	11	V
<input type="checkbox"/>	<input type="checkbox"/>	12	V
<input type="checkbox"/>	<input type="checkbox"/>	13	F
<input type="checkbox"/>	<input type="checkbox"/>	14	V

Apellidos, Nombre:
DNI:

Examen PED abril 2013

Modalidad 1

Normas:

- * Tiempo para efectuar el test: **25 minutos**.
- * Una pregunta mal contestada elimina una correcta.
- * Las soluciones al examen se dejarán en el campus virtual.
- * **Una vez empezado el examen no se puede salir del aula hasta finalizarlo.**
- * En la hoja de contestaciones el verdadero se corresponderá con la **A**, y el falso con la **B**.

	V	F	
En C++, después de invocar el destructor (~NombreClase) de un objeto, no se puede acceder a los miembros (propiedades y métodos) de dicho objeto.	<input type="checkbox"/>	<input type="checkbox"/>	1 F
Sea un vector de números naturales. La operación <i>eliminar</i> que borra las posiciones pares del vector marcándolas con “0”, vista en clase, se define así: eliminar: vector -> vector Var v:vector; i: entero; x:natural; eliminar(crear()) = crear() si (i MOD 2) == 0 entonces eliminar(asignar(v,i,x)) = asignar(eliminar(v),i,x) si no eliminar(asignar(v,i,x)) = asignar(eliminar(v),i,0)	<input type="checkbox"/>	<input type="checkbox"/>	2 F
En C++, el puntero this sólo se puede usar dentro de los métodos de la clase.	<input type="checkbox"/>	<input type="checkbox"/>	3 V
En general, las operaciones modificadoras y consultoras se especifican en términos de las generadoras. En ocasiones, una operación modificadora puede especificarse en términos de otras modificadoras o consultoras. Diremos que se trata de una operación derivada.	<input type="checkbox"/>	<input type="checkbox"/>	4 V
En la escala de complejidades, la mejor complejidad temporal que se puede conseguir en un algoritmo es O(n), siendo “n” la talla del problema.	<input type="checkbox"/>	<input type="checkbox"/>	5 F
El algoritmo de búsqueda binaria estudiado en clase (búsqueda de un elemento en un vector ordenado) tiene una complejidad de $\Omega(1)$.	<input type="checkbox"/>	<input type="checkbox"/>	6 V
La operación <i>BorrarItem</i> tiene la siguiente sintaxis y semántica: <i>BorrarItem</i> : LISTA, ITEM -> LISTA <i>BorrarItem</i> (Crear, i) = Crear <i>BorrarItem</i> (IC(L1,j), i) = si (i == j) entonces L1 sino IC (<i>BorrarItem</i> (L1, i), j)	<input type="checkbox"/>	<input type="checkbox"/>	7 V
Esta operación borra la primera ocurrencia del item que se encuentra en la lista.	<input type="checkbox"/>	<input type="checkbox"/>	8 F
La operación <i>base</i> , vista en clase, que actúa sobre una pila y devuelve la base de la pila (el primer elemento que se ha apilado) es la siguiente: base(pila) -> item Var p: pila; x: item; base(crear()) = error() base(apilar(crear(),x)) = x base(apilar(p,x)) = base(desapilar(p))	<input type="checkbox"/>	<input type="checkbox"/>	9 V
El número mínimo de nodos que tiene un árbol AVL de altura 4 es 7.	<input type="checkbox"/>	<input type="checkbox"/>	10 F
El grado de un nodo es el número máximo de items asociados a dicho nodo.	<input type="checkbox"/>	<input type="checkbox"/>	11 V
El coste temporal (en su peor caso) de insertar una etiqueta en un árbol binario de búsqueda es lineal respecto al número de nodos del árbol.	<input type="checkbox"/>	<input type="checkbox"/>	12 V
Dada la siguiente representación secuencial del árbol binario A, [14 8 19 5] el elemento 5 es el hijo izquierda del elemento 8.	<input type="checkbox"/>	<input type="checkbox"/>	13 F
Cuando se realiza un borrado en un árbol AVL, en el camino de vuelta atrás para actualizar los factores de equilibrio, como mucho sólo se va a efectuar una rotación.	<input type="checkbox"/>	<input type="checkbox"/>	14 F
Dado un único recorrido de un árbol binario, es posible reconstruir dicho árbol.	<input type="checkbox"/>	<input type="checkbox"/>	

Apellidos, Nombre:
DNI:

Examen PED abril 2013 Modalidad 2

Normas:

- * Tiempo para efectuar el test: **25 minutos**.
- * Una pregunta mal contestada elimina una correcta.
- * Las soluciones al examen se dejarán en el campus virtual.
- * **Una vez empezado el examen no se puede salir del aula hasta finalizarlo.**
- * En la hoja de contestaciones el verdadero se corresponderá con la **A**, y el falso con la **B**.

	V	F	
La operación <i>base</i> , vista en clase, que actúa sobre una pila y devuelve la base de la pila (el primer elemento que se ha apilado) es la siguiente: base(pila) -> item Var p: pila; x: item; base(crear()) = error() base(apilar(crear(),x)) = x base(apilar(p,x)) = base(desapilar(p))	<input type="checkbox"/>	<input type="checkbox"/>	1 F
Sea un vector de números naturales. La operación <i>eliminar</i> que borra las posiciones pares del vector marcándolas con “0”, vista en clase, se define así: eliminar: vector -> vector Var v:vector; i: entero; x:natural; eliminar(crear()) = crear() si (i MOD 2) == 0 entonces eliminar(asignar(v,i,x)) = asignar(eliminar(v),i,x) si no eliminar(asignar(v,i,x)) = asignar(eliminar(v),i,0)	<input type="checkbox"/>	<input type="checkbox"/>	2 F
El algoritmo de búsqueda binaria estudiado en clase (búsqueda de un elemento en un vector ordenado) tiene una complejidad de $\Omega(1)$.	<input type="checkbox"/>	<input type="checkbox"/>	3 V
En C++, después de invocar el destructor (~NombreClase) de un objeto, no se puede acceder a los miembros (propiedades y métodos) de dicho objeto.	<input type="checkbox"/>	<input type="checkbox"/>	4 F
En la escala de complejidades, la mejor complejidad temporal que se puede conseguir en un algoritmo es O(n), siendo “n” la talla del problema.	<input type="checkbox"/>	<input type="checkbox"/>	5 F
En C++, el puntero this sólo se puede usar dentro de los métodos de la clase.	<input type="checkbox"/>	<input type="checkbox"/>	6 V
La operación <i>BorrarItem</i> tiene la siguiente sintaxis y semántica: BorrarItem: LISTA, ITEM -> LISTA BorrarItem(Crear, i) = Crear BorrarItem(IC(L1,j), i) = si (i == j) entonces L1 sino IC (BorrarItem (L1, i), j)	<input type="checkbox"/>	<input type="checkbox"/>	7 V
Esta operación borra la primera ocurrencia del item que se encuentra en la lista.	<input type="checkbox"/>	<input type="checkbox"/>	8 V
En general, las operaciones modificadoras y consultoras se especifican en términos de las generadoras. En ocasiones, una operación modificadora puede especificarse en términos de otras modificadoras o consultoras. Diremos que se trata de una operación derivada.	<input type="checkbox"/>	<input type="checkbox"/>	9 F
Dado un único recorrido de un árbol binario, es posible reconstruir dicho árbol.	<input type="checkbox"/>	<input type="checkbox"/>	10 F
Cuando se realiza un borrado en un árbol AVL, en el camino de vuelta atrás para actualizar los factores de equilibrio, como mucho sólo se va a efectuar una rotación.	<input type="checkbox"/>	<input type="checkbox"/>	11 V
El coste temporal (en su peor caso) de insertar una etiqueta en un árbol binario de búsqueda es lineal respecto al número de nodos del árbol.	<input type="checkbox"/>	<input type="checkbox"/>	12 V
Dada la siguiente representación secuencial del árbol binario A, [14 8 19 5] el elemento 5 es el hijo izquierda del elemento 8.	<input type="checkbox"/>	<input type="checkbox"/>	13 F
El grado de un nodo es el número máximo de items asociados a dicho nodo.	<input type="checkbox"/>	<input type="checkbox"/>	14 V

Apellidos, Nombre:

DNI:

Examen PED abril 2014

Modalidad 0

Normas:

- Tiempo para efectuar el test: **20 minutos**.
- Una pregunta mal contestada elimina una correcta.
- Las soluciones al examen se dejarán en el campus virtual.
- **Una vez empezado el examen no se puede salir del aula hasta finalizarlo.**
- En la hoja de contestaciones el verdadero se corresponderá con la **A**, y el falso con la **B**.

	V	F	
La operación <i>palindromo</i> (sobre un vector) vista en clase es la siguiente: Var v: vector; i,x: natural; <i>palindromo(crear_vector()) = VERDADERO</i> <i>palindromo(asig(v,i,x)) = si i<= 50</i> entonces si recu(v,100-i+1) == x entonces palindromo(asig(v,i,x)) sino FALSO sino VERDADERO	<input type="checkbox"/>	<input type="checkbox"/>	1 F
Una operación del TAD X que tenga la sintaxis Crear() →X es una operación constructora generadora.	<input type="checkbox"/>	<input type="checkbox"/>	2 V
En C++, al hacer layering los métodos de la clase derivada pueden acceder a la parte pública y privada de la clase base.	<input type="checkbox"/>	<input type="checkbox"/>	3 F
En C++, si un objeto se sale de ámbito entonces se invoca automáticamente al destructor de ese objeto.	<input type="checkbox"/>	<input type="checkbox"/>	4 V
El algoritmo de intercambio directo o burbuja estudiado en clase (ordenación de los elementos de un vector) tiene una complejidad promedio de $\Theta(n^2)$, siendo n el número de elementos del vector.	<input type="checkbox"/>	<input type="checkbox"/>	5 V
La complejidad espacial es la cantidad de recursos espaciales que un algoritmo consume o necesita para su ejecución.	<input type="checkbox"/>	<input type="checkbox"/>	6 V
La complejidad temporal en el peor caso para la inserción de un elemento en una lista ordenada y en otra no ordenada, que no permiten elementos repetidos, siempre es lineal con el número de elementos en ambos casos.	<input type="checkbox"/>	<input type="checkbox"/>	7 V
El tipo de datos vector (visto en clase) se define como un conjunto en el que sus componentes ocupan posiciones consecutivas de memoria.	<input type="checkbox"/>	<input type="checkbox"/>	8 F
Sea el TIPO arbin definido en clase. La semántica de la operación nodos es la siguiente: Var i,d:arbin; x:item; <i>nodos(crear_arbin())=0</i> <i>nodos(eraizar(i,x,d))=nodos(i)+nodos(d)</i>	<input type="checkbox"/>	<input type="checkbox"/>	9 F
Es posible reconstruir un único árbol binario de altura 6 a partir de un recorrido en postorden con 63 etiquetas.	<input type="checkbox"/>	<input type="checkbox"/>	10 V
El ítem medio (según la relación de orden) almacenado en un árbol binario de búsqueda lleno siempre se encuentra en la raíz.	<input type="checkbox"/>	<input type="checkbox"/>	11 V
A los árboles generales también se les llama árboles multicamino de búsqueda.	<input type="checkbox"/>	<input type="checkbox"/>	12 F
Las rotaciones que hay que realizar en los árboles AVL para mantenerlos balanceados tienen un coste temporal (en su peor caso) lineal con respecto al número de items del árbol.	<input type="checkbox"/>	<input type="checkbox"/>	13 F
El grado de los árboles AVL puede ser +1, 0 ó -1.	<input type="checkbox"/>	<input type="checkbox"/>	14 F

Apellidos, Nombre:
DNI:

Examen PED marzo 2015

Modalidad 0

Normas:

- Tiempo para efectuar el test: **20 minutos**.
- Una pregunta mal contestada elimina una correcta.
- Las soluciones al examen se dejarán en el campus virtual.
- **Una vez empezado el examen no se puede salir del aula hasta finalizarlo.**
- En la hoja de contestaciones el verdadero se corresponderá con la **A**, y el falso con la **B**.

	V	F		
En la especificación algebraica, las operaciones constructoras se clasifican en generadoras y modificadoras.	<input type="checkbox"/>	<input type="checkbox"/>	1	V
Las ecuaciones (vistas en clase) que permiten realizar la suma de números naturales son las siguientes: VAR x, y: natural; suma(x, cero) = x suma(cero, x) = x suma(x, suc(y)) = suma(x, y)	<input type="checkbox"/>	<input type="checkbox"/>	2	F
Dentro de la especificación algebraica de los números naturales, definimos la sintaxis de la función F como: F: natural → BOOL, y su semántica como: F(cero)=TRUE, F(suc(cero))=FALSE, F(suc(suc(x)))=F(x). Para el número natural x=35, la función F devolvería FALSE. Nota: se asume que x=35 es la forma simplificada de indicar x=suc(suc(.....suc(cero).....))).	<input type="checkbox"/>	<input type="checkbox"/>	3	V
Todo árbol binario de altura 9 y 511 nodos es un árbol binario lleno y además es árbol binario de búsqueda.	<input type="checkbox"/>	<input type="checkbox"/>	4	F
Sea el método Primera perteneciente a la clase TLista que devuelve la primera posición de la lista que lo invoca:	<input type="checkbox"/>	<input type="checkbox"/>	5	F
TPosicion TLista::Primera()	<input type="checkbox"/>	<input type="checkbox"/>	6	F
{ TPosicion p; p.pos = primero; return p; }	<input type="checkbox"/>	<input type="checkbox"/>	7	V
En el método Primera, se invoca de forma implícita a los constructores de TPosicion y TLista.	<input type="checkbox"/>	<input type="checkbox"/>	8	F
En C++, si la variable p es un puntero a un objeto, entonces la expresión p.f() es sintácticamente correcta.	<input type="checkbox"/>	<input type="checkbox"/>	9	V
La complejidad temporal del siguiente fragmento de código es O(n)	<input type="checkbox"/>	<input type="checkbox"/>	10	V
int i, j, n, sum; for (i = 4; i < n; i++) { for (j = i-3, sum = a[i-4]; j <= i; j++) sum += a[j]; cout << "La suma del subarray " << i-4 << " es " << sum << endl; }	<input type="checkbox"/>	<input type="checkbox"/>	11	V
La mejor complejidad temporal que se puede conseguir en un algoritmo es O(n), siendo "n" la talla del problema.	<input type="checkbox"/>	<input type="checkbox"/>	12	F
Es posible reconstruir un único árbol binario de búsqueda a partir de su recorrido en postorden	<input type="checkbox"/>	<input type="checkbox"/>	13	V
El máximo número de nodos en un nivel i-1 de un árbol binario es 2^{i-2} , $i \geq 2$	<input type="checkbox"/>	<input type="checkbox"/>	14	F
Un camino en un árbol es una secuencia a_1, \dots, a_s de árboles tal que para todo $i \in \{1, \dots, s-1\}$, a_{i+1} es subárbol de a_i .				
El grado de un árbol es el máximo nivel que pueden tener sus subárboles.				
La operación desencolar vista en clase es la siguiente:				
VAR c: cola, x: item; desencolar(crear()) = crear() si esvacia(c) entonces desencolar(encolar(c, x)) = crear() si no desencolar(encolar(c, x)) = encolar(desencolar(c), x)				
El ítem medio (según la relación de orden en la búsqueda) almacenado en un árbol binario de búsqueda siempre se encuentra en la raíz.				

TESTS DE LOS EXÁMENES

V	F	
<input type="checkbox"/>	<input type="checkbox"/>	V 1.
<input type="checkbox"/>	<input type="checkbox"/>	V 2.
<input type="checkbox"/>	<input type="checkbox"/>	F 3.
<input type="checkbox"/>	<input type="checkbox"/>	F 4.
<input type="checkbox"/>	<input type="checkbox"/>	F 5.
<input type="checkbox"/>	<input type="checkbox"/>	V 6.
<input type="checkbox"/>	<input type="checkbox"/>	V 7.
<input type="checkbox"/>	<input type="checkbox"/>	V 8.
<input type="checkbox"/>	<input type="checkbox"/>	V 9.
<input type="checkbox"/>	<input type="checkbox"/>	V 10.
<input type="checkbox"/>	<input type="checkbox"/>	F 11.
<input type="checkbox"/>	<input type="checkbox"/>	V 12.
<input type="checkbox"/>	<input type="checkbox"/>	F 13.
<input type="checkbox"/>	<input type="checkbox"/>	V 14.

V	F		
<input type="checkbox"/>	<input type="checkbox"/>	F	1.
<input type="checkbox"/>	<input type="checkbox"/>	F	2.
<input type="checkbox"/>	<input type="checkbox"/>	F	3.
<input type="checkbox"/>	<input type="checkbox"/>	F	4.
<input type="checkbox"/>	<input type="checkbox"/>	F	5.
<input type="checkbox"/>	<input type="checkbox"/>	V	6.
<input type="checkbox"/>	<input type="checkbox"/>	V	7.
<input type="checkbox"/>	<input type="checkbox"/>	F	8.
<input type="checkbox"/>	<input type="checkbox"/>	F	9.
<input type="checkbox"/>	<input type="checkbox"/>	V	10.
<input type="checkbox"/>	<input type="checkbox"/>	V	11.
<input type="checkbox"/>	<input type="checkbox"/>	V	12.
<input type="checkbox"/>	<input type="checkbox"/>	F	13.
<input type="checkbox"/>	<input type="checkbox"/>	F	14.
<input type="checkbox"/>	<input type="checkbox"/>	V	15.

La operación crear_pila() es constructora modificadora.

En C++, una forma correcta de copiar una cadena es la siguiente:

```
char a[50] = "Tipos Abstractos de Datos";
char *b;
b = new char[strlen(a)];
strcpy(b, a);
```

El caso peor de la búsqueda es más eficiente en una lista ordenada que en una lista cuyos elementos no están ordenados.

En un árbol binario cada elemento puede tener como máximo dos predecesores.

Si se implementa el algoritmo de ordenación de un vector “heapsort” utilizando un heap máximo los elementos quedan ordenados en el vector de forma descendente.

Dado un recorrido en preorder (RID) de un árbol AVL es posible reconstruir un único árbol AVL.

El número máximo de elementos que se puede almacenar en un árbol 2-3 de altura h es $3^h - 1$

Al borrar un elemento en un árbol 2-3-4 se puede realizar una operación de DIVIDERAZ.

En un árbol B m-camino de búsqueda con m=16: en cualquier nodo excepto la raíz hay 8 ítems como mínimo.

La especificación algebraica de la siguiente operación permite la inserción de claves repetidas (C: ConjuntoConClavesRepetidas; x, y: Ítem):

$$\text{Insertar}(\text{Insertar}(C, x), y) \Leftrightarrow \text{Insertar}(\text{Insertar}(C, y), x)$$

En el TAD Diccionario con dispersión cerrada, con función de redispersión “ $h(x) = (H(x) + k(x)*i) \text{ MOD } B$ ”, con B=6 se puede dar la situación de que en una búsqueda no se acceda a todas las posiciones de la tabla.

El siguiente vector representa un montículo máximo:

10	5	3	1	2
----	---	---	---	---

Para todo nodo de un árbol Leftist, se cumple que la altura de su hijo izquierdo es menor que la de su hijo derecho.

La complejidad temporal de la búsqueda en un trie es lineal respecto al número de palabras almacenadas

Un bosque extendido en profundidad de un grafo no dirigido es un grafo acíclico.

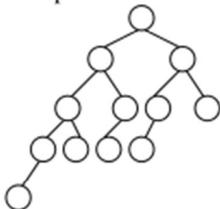
V	F	
<input type="checkbox"/>	<input type="checkbox"/>	1. V
<input type="checkbox"/>	<input type="checkbox"/>	2. F
<input type="checkbox"/>	<input type="checkbox"/>	3. F
<input type="checkbox"/>	<input type="checkbox"/>	4. F
<input type="checkbox"/>	<input type="checkbox"/>	5. F
<input type="checkbox"/>	<input type="checkbox"/>	6. V
<input type="checkbox"/>	<input type="checkbox"/>	7. F
<input type="checkbox"/>	<input type="checkbox"/>	8. F
<input type="checkbox"/>	<input type="checkbox"/>	9. F
<input type="checkbox"/>	<input type="checkbox"/>	10. V
<input type="checkbox"/>	<input type="checkbox"/>	11. V
<input type="checkbox"/>	<input type="checkbox"/>	12. V
<input type="checkbox"/>	<input type="checkbox"/>	13. F
<input type="checkbox"/>	<input type="checkbox"/>	14. F

V	F	
<input type="checkbox"/>	<input type="checkbox"/>	1. F
<input type="checkbox"/>	<input type="checkbox"/>	2. F
<input type="checkbox"/>	<input type="checkbox"/>	3. F
<input type="checkbox"/>	<input type="checkbox"/>	4. V
<input type="checkbox"/>	<input type="checkbox"/>	5. V
<input type="checkbox"/>	<input type="checkbox"/>	6. V
<input type="checkbox"/>	<input type="checkbox"/>	7. F
<input type="checkbox"/>	<input type="checkbox"/>	8. F
<input type="checkbox"/>	<input type="checkbox"/>	9. F
<input type="checkbox"/>	<input type="checkbox"/>	10. V
<input type="checkbox"/>	<input type="checkbox"/>	11. V
<input type="checkbox"/>	<input type="checkbox"/>	12. V
<input type="checkbox"/>	<input type="checkbox"/>	13. F
<input type="checkbox"/>	<input type="checkbox"/>	14. F

Siempre que se realiza una rotación DD en el borrado de un elemento en un árbol AVL decrece la altura del subárbol sobre el que se realiza la rotación.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	1	F
Un árbol 2-3 es un árbol m -camino de búsqueda con $m=3$.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	2	V
La complejidad temporal (en su peor caso) de buscar un elemento en un vector ordenado utilizando un algoritmo de búsqueda binaria es $\alpha(\log_2)$ [siendo n el número de elementos].	<input type="checkbox"/>	<input checked="" type="checkbox"/>	3	F
Un árbol rojo-negro, en el que no hay ningún enlace rojo, es un árbol binario lleno	<input type="checkbox"/>	<input checked="" type="checkbox"/>	4	V
En una tabla de dispersión cerrada con la siguiente función de redispersión para la clave 14: $h_i(14)=(28 + 7*i) \text{ MOD } 2000$, se recorrerán todas las posiciones de la tabla buscando una posición libre.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	5	V
En el algoritmo HeapSort, después del primer paso de insertar todos los elementos en un Heap representado como un vector, los elementos del mismo quedan ordenados.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	6	F
Un bosque extendido en profundidad de un grafo dirigido al que se le añaden los arcos de avance y de cruce es un grafo acíclico dirigido.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	7	V
Sea $G=(V,A)$ un grafo dirigido. Diremos que $G''=(V'',A'')$ es un árbol extendido de $G \Leftrightarrow V''=V, A'' \subset A, \forall v \in V'' \Rightarrow \text{grado}_E(v) \leq 1$.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	8	V
La operación <i>BorrarItem</i> , que borra la primera ocurrencia del ítem i que se encuentre en la lista, tiene la siguiente sintaxis y semántica:	<input type="checkbox"/>	<input checked="" type="checkbox"/>	9	F
	BorrarItem: LISTA, ITEM -> LISTA BorrarItem(Crear, i) = Crear BorrarItem(IC(L1,j), i) = si (i == j) entonces BorrarItem (L1, i) sino IC (BorrarItem (L1, i), j)			
Se puede reconstruir un único árbol binario de búsqueda teniendo sus recorridos en preorden y postorden.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	10	V
La estructura de datos árbol aparece porque los elementos que lo constituyen mantienen una estructura jerárquica, obtenida a partir de estructuras lineales, al eliminar el requisito de que cada elemento tiene como máximo un predecesor.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	11	F
Al realizar un recorrido en inorder de un montículo obtenemos una sucesión de claves ordenadas.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	12	F
La función de redispersión en una tabla hash abierta tiene que cumplir como condición para que se recorran todas las posiciones del vector que el valor de B sea primo	<input type="checkbox"/>	<input checked="" type="checkbox"/>	13	F
Un grafo que tiene componentes fuertemente conexas es un grafo necesariamente libre de ciclos	<input type="checkbox"/>	<input checked="" type="checkbox"/>	14	F

V F
 1. V

El siguiente árbol está balanceado con respecto a la altura

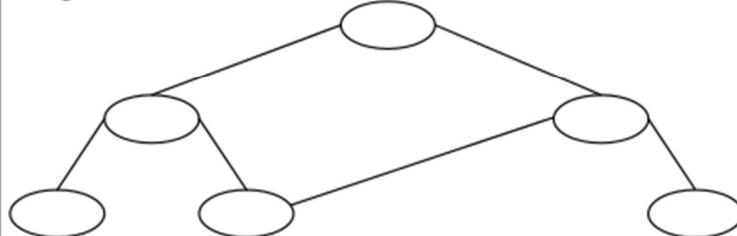


2. V
 3. F
 4. F

La siguiente función de C++, `int& Incremento(int valor){valor=valor+5;return valor;}`, devuelve el resultado por referencia.

En las colas, las inserciones y borrados se realizan por el mismo extremo.

La siguiente estructura es un árbol binario:



5. F
 6. V
 7. F
 8. F
 9. V
 10. F
 11. V
 12. F
 13. F
 14. V

Un árbol completo es un árbol completamente equilibrado

Los árboles AVL son árboles balanceados con respecto a la altura de los subárboles.

En un árbol 2-3, la diferencia en número de nodos entre los subárboles de la raíz es como mucho 1.

Un árbol rojo-negro, en el que no hay ningún enlace rojo, es un árbol binario completo.

Un árbol binario de búsqueda con altura 7 y 127 nodos es un árbol B con m=2

En un árbol m-camino de búsqueda, todos los nodos excepto la raíz tienen al menos $m/2$ hijos.

En la dispersión cerrada se pueden producir colisiones entre claves no sinónimas

En un montículo doble, un elemento "j" del montículo máximo es el simétrico de un único elemento "i" del montículo mínimo.

Un árbol Rojo-Negro cumple las propiedades de un árbol Leftist.

Al representar un grafo no dirigido con una matriz de adyacencia, su diagonal principal (casillas i,i) siempre tendrá valores Falso.

EN C++, la instrucción `cola a = b;` donde `b` es de tipo `cola` invoca al constructor de copia.

1. v
 2. V

La semántica de la operación `concatena` del tipo cola vista en clase es la siguiente:

```
VAR c, q: cola; x: item;
concatena( c, crear_cola() ) = c
concatena(crear_cola(), c) = c
concatena( c, encolar( q, x ) ) = encolar( concatena( c, q ), x )
```

Se puede reconstruir un árbol binario cualquiera teniendo sus recorridos en preorden e inorden.

En la operación de borrado de un elemento en un árbol AVL, si se realiza una rotación doble siempre decrece la altura del árbol sobre el que se ha realizado la rotación.

El número máximo de elementos que se puede almacenar en un árbol 2-3 de altura h coincide con el número de elementos que hay en un árbol binario lleno de altura h .

En un árbol 2-3-4, los nodos pueden tener 1, 2 ó 3 hijos.

Todo árbol Rojo - Negro es un árbol 2-3-4.

Todo árbol B con $m=4$ es un árbol 2-3

El siguiente vector representa un montículo máximo:

10	5	(Vacío)	1	2
----	---	---------	---	---

3. V
 4. V
 5. F
 6. F
 7. F
 8. F
 9. F

Un bosque extendido en profundidad de un grafo dirigido al que se le añaden los arcos de avance es un grafo acíclico dirigido.

10. V

	V	F
Las operaciones constructoras generadoras de un tipo permiten obtener cualquier valor de dicho tipo.	<input type="checkbox"/>	<input type="checkbox"/> 1. V
En C++, si no se ha implementado la sobrecarga del operador asignación, se invoca automáticamente al constructor de copia.	<input type="checkbox"/>	<input type="checkbox"/> 2. F
Es posible reconstruir un único árbol binario de altura 6 a partir de un recorrido en postorden con 63 etiquetas.	<input type="checkbox"/>	<input type="checkbox"/> 3. V
La semántica de la operación nodos del tipo <i>arbin</i> vista en clase es la siguiente:	<input type="checkbox"/>	<input type="checkbox"/> 4. F
<i>VAR i, d: arbin; x: item;</i> <i>nodos(crea_arbin()) = 0</i> <i>nodos(enraizar(i, x, d)) = nodos(i) + nodos(d)</i>		
Se puede reconstruir un único árbol binario cualquiera teniendo sus recorridos en preorden y postorden.	<input type="checkbox"/>	<input type="checkbox"/> 5. F
La semántica de la operación <i>recu</i> vista en clase es la siguiente:	<input type="checkbox"/>	<input type="checkbox"/> 6. F
<i>VAR v: vector; i, j: int; x: item;</i> <i>recu(crear_vector(), i) = error_item()</i> <i>recu(asig(v, i, x), j)</i> <i>si (i == j) entonces x</i> <i>sino FALSO</i> <i>fsi</i>		
En un árbol AVL cuyo factor de equilibrio es -2, al insertar un elemento en la rama derecha, el árbol vuelve al estado de equilibrio.	<input type="checkbox"/>	<input type="checkbox"/> 7.
Dado un árbol 2-3 de altura h con n ítems con todos sus nodos del tipo 2-Nodo: la complejidad de la operación de búsqueda de un ítem es $O(\log_2 n)$	<input type="checkbox"/>	<input type="checkbox"/> 8. V
Un árbol binario de búsqueda lleno de altura 4 es un árbol 2-3-4, pero no se puede conseguir a partir de un árbol inicialmente vacío y utilizando las operaciones de inserción y borrado de un árbol 2-3-4	<input type="checkbox"/>	<input type="checkbox"/> 9. V
Un grafo no dirigido puede tener aristas que empiecen y acaben en el mismo vértice.	<input type="checkbox"/>	<input type="checkbox"/> 10. F
El siguiente árbol es leftist mínimo:		<input type="checkbox"/> 11. V
<pre> graph TD 1((1)) --- 7((7)) 1 --- 3((3)) 3 --- 4((4)) </pre>		
Un trie cumple las propiedades de un árbol general.	<input type="checkbox"/>	<input type="checkbox"/> 12. V

V	F		
<input type="checkbox"/>	<input checked="" type="checkbox"/>	1	F
Las ecuaciones (vistas en clase) que permiten realizar la suma de números naturales son las siguientes: VAR x, y: natural; suma(x, cero) = x suma(cero, x) = x suma(x, suc(y)) = suma(suc(x), y)			
<input type="checkbox"/>	<input checked="" type="checkbox"/>	2	F
<input type="checkbox"/>	<input checked="" type="checkbox"/>	3	F
En C++, la siguiente declaración es INCORRECTA : const int& a = 1;			
Dadas las clases TDir y TVectorDir con todos sus métodos implementados: constructor, constructor sobrecargado, sobrecarga del corchete, sobrecarga del operador salida (se muestra el contenido de cada posición del vector dejando un espacio), etc. Nota: El constructor por defecto crea un vector vacío.			
El constructor a partir de una dimensión, pone todos los elementos a 0.			
<pre>class TDir class TVectorDir main() { public: public: TDir a(1,1); private: private: TVectorDir v; int e1; TDir *vector; cout<<"v_anter= "<<v<<endl; int e2; int longitud; v[1]=a; }; }; }; cout<<"v_despues= "<<v<<endl; }</pre>			
El resultado obtenido tras la ejecución del <i>main()</i> sería: v_anter= 0 0 v_despues= 1 1			
<input type="checkbox"/>	<input checked="" type="checkbox"/>	4	V
<input type="checkbox"/>	<input checked="" type="checkbox"/>	5	F
<input type="checkbox"/>	<input checked="" type="checkbox"/>	6	V
La complejidad espacial es la cantidad de recursos espaciales que un algoritmo consume o necesita para su ejecución.			
La complejidad temporal de un algoritmo depende de la complejidad espacial del mismo.			
La semántica de la operación <i>desencolar</i> vista en clase es la siguiente:			
<pre>VAR c: cola, x: item; si esvacia(c) entonces desencolar(encolar(c, x)) = crear_cola () si no desencolar(encolar(c, x)) = encolar(desencolar(c), x)</pre>			
<input type="checkbox"/>	<input checked="" type="checkbox"/>	7	V
<input type="checkbox"/>	<input checked="" type="checkbox"/>	8	F
<input type="checkbox"/>	<input checked="" type="checkbox"/>	9	V
<input type="checkbox"/>	<input checked="" type="checkbox"/>	10	V
<input type="checkbox"/>	<input checked="" type="checkbox"/>	11	F
Un árbol con un único nodo es un árbol lleno.			
Un árbol con un único nodo tiene un único camino cuya longitud es 1.			
Dados los recorridos de preorden, postorden y niveles de un árbol binario de altura 7 y 64 hojas es posible reconstruir un único árbol binario.			
En la representación de conjuntos mediante listas, la complejidad espacial es proporcional al tamaño del conjunto representado.			
En un grafo dirigido pueden existir infinitas aristas para un número "n" de vértices.			

V	F		
<input type="checkbox"/>	<input type="checkbox"/>	1	V
<input type="checkbox"/>	<input type="checkbox"/>	2	F
<input type="checkbox"/>	<input type="checkbox"/>	3	F
<input type="checkbox"/>	<input type="checkbox"/>	4	V
<input type="checkbox"/>	<input type="checkbox"/>	5	F
<input type="checkbox"/>	<input type="checkbox"/>	6	V
<input type="checkbox"/>	<input type="checkbox"/>	7	V
<input type="checkbox"/>	<input type="checkbox"/>	8	V
<input type="checkbox"/>	<input type="checkbox"/>	9	F
<input type="checkbox"/>	<input type="checkbox"/>	10	V
<input type="checkbox"/>	<input type="checkbox"/>	11	F
<input type="checkbox"/>	<input type="checkbox"/>	12	V
<input type="checkbox"/>	<input type="checkbox"/>	13	V

La complejidad logarítmica aparece en algoritmos que descartan muchos valores (generalmente la mitad) en un único paso.

Dada la sintaxis de la función $IC(lista,item) \rightarrow lista$, que inserta un elemento a la cabeza de la lista pasada como parámetro y $crear() \rightarrow lista$, que crea una lista vacía. La siguiente secuencia: $IC(IC(crear(),a),b),c$, daría como resultado una lista con los elementos en este orden: $a \rightarrow b \rightarrow c$, donde a es el primer elemento de la lista

Dentro de la especificación algebraica de los números naturales definimos la sintaxis de la función F como: $F: natural \rightarrow BOOL$, y su semántica como: $F(cero)=TRUE$, $F(suc(cero))=FALSE$, $F(suc(suc(x)))=F(x)$. Para el número natural $x=35$, la función F devolvería TRUE.

En C++, si un objeto se sale de ámbito entonces se invoca automáticamente al destructor de ese objeto.

En C++, el constructor de copia recibe como argumento un objeto del mismo tipo pasado por referencia o por valor.

El algoritmo de búsqueda binaria estudiado en clase (búsqueda de un elemento en un vector ordenado) tiene una complejidad de $\Omega(1)$.

La complejidad espacial es la cantidad de recursos espaciales que un algoritmo consume o necesita para su ejecución

La operación BorrarItem, que borra todas las ocurrencias del item i que se encuentren en la lista, tiene la siguiente sintaxis y semántica:

```

BorrarItem: LISTA, ITEM -> LISTA
BorrarItem( Crear, i) =      Crear
BorrarItem( IC(L1,j), i) =    si ( i == j ) entonces BorrarItem ( L1, i )
                                sino IC ( BorrarItem ( L1, i ), j )

```

La semántica de la operación obtener en una lista con acceso por posición es la siguiente ($IC=InsertarCabeza(Lista, Ítem)$, p : posición, $l1$: lista, x : ítem):

```

obtener(crear(),p)=error_item()
si p == primera(IC(l1,x)) entonces obtener(IC(l1,x),p)=x
sino obtener(IC(l1,x),p)=IC(obtener(l1,p),x)

```

El máximo número de nodos en un nivel $i-1$ de un árbol binario es 2^{i-2} , $i \geq 2$

Sea el TIPO arbin definido en clase. La semántica de la operación nodos es la siguiente:

```

Var i,d:arbin; x:item;
nodos(crear_arbin())=0
nodos(enraizar(i,x,d))=nodos(i)+nodos(d)

```

El coste temporal en su peor caso de insertar una etiqueta en un árbol binario de búsqueda es lineal con la altura del árbol

Se puede obtener un único árbol 2-3-4 a partir de su recorrido por niveles

	<input type="checkbox"/>	<input checked="" type="checkbox"/>	V	F	
Las ecuaciones (vistas en clase) que permiten realizar la multiplicación de números naturales son las siguientes:	<input type="checkbox"/>	<input checked="" type="checkbox"/>	1	V	
<pre>VAR x, y: natural; mult(cero, x) = cero mult(x, cero) = cero mult(suc(y), x) = suma(mult(y, x), x)</pre>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	2	F	
En C++ y cuando se emplea composición (layering), los métodos de la clase derivada pueden acceder a la parte protegida de la clase base.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	3	F	
Para el siguiente algoritmo, la complejidad sería $O(n^2)$:	<input type="checkbox"/>	<input checked="" type="checkbox"/>	4	F	
<pre>for (i=0; i<100; i++) for (j=0; j<100; j++) if (v[i]<v[j]) v[i] = v[j]; else v[j] = v[i];</pre>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	5	V	
Dados los recorridos de preorder, postorden y niveles de un árbol binario sólo se puede reconstruir un único árbol binario.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	6	F	
Sabiendo que A es un árbol binario de búsqueda completo y dado su recorrido inorder 1,4,6,7,9,12,14,20,23. La secuencia 12,7,20,4,9,14,23,1,6, se corresponde con su recorrido por niveles.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	V	F	
<i>Cuando se borra un elemento en un AVL habrá cascos en los que no sea necesario reestructurar</i>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	1	F	
Sea el método Primera perteneciente a la clase TLista que devuelve la primera posición de la lista que lo invoca:	<input type="checkbox"/>	<input checked="" type="checkbox"/>	2	V	
<pre>TPosicion TLista::Primera() { TPosicion p; p.pos = lis; return p; }</pre>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	3	V	
En la línea resaltada, se invoca a la sobrecarga del operador asignación entre objetos del tipo TPosicion.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	4	F	
En la escala de complejidades se cumple que $O(n \log n) \subset O(n^2)$.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	5	F	
La operación BorrarItem tiene la siguiente sintaxis y semántica:	<input type="checkbox"/>	<input checked="" type="checkbox"/>	6	V	
BorrarItem: LISTA, ITEM -> LISTA BorrarItem(Crear, i) = Crear BorrarItem(IC(L1,j), i) = si (i == j) entonces L1 sino IC (BorrarItem (L1, i), j)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	7	F	
Esta operación borra la primera ocurrencia del item que se encuentra en la lista	<input type="checkbox"/>	<input checked="" type="checkbox"/>	8	F	
El nivel de un nodo en un árbol coincide con la longitud del camino desde la raíz a dicho nodo	<input type="checkbox"/>	<input checked="" type="checkbox"/>	9	F	
A los árboles generales también se les llama árboles multicamino de búsqueda	<input type="checkbox"/>	<input checked="" type="checkbox"/>	10	F	
Cuando se realiza una inserción en un AVL, en el camino de vuelta atrás para actualizar los factores de equilibrio, sólo se va a efectuar una rotación como mucho	<input type="checkbox"/>	<input checked="" type="checkbox"/>	11	F	
Dado un árbol 2-3 con n items con todos sus nodos del tipo 2-Nodo: la complejidad de la operación de búsqueda de un ítem es $O(\log_2 n)$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	12	V	
Un árbol Rojo-Negro es una representación sobre árbol binario de un árbol 2-3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	13	F	
Un árbol rojo-negro es un árbol binario balanceado respecto a la altura	<input type="checkbox"/>	<input checked="" type="checkbox"/>	14	V	
La raíz del árbol B m-camino de búsqueda siempre tiene al menos $m/2$ claves o etiquetas	<input type="checkbox"/>	<input checked="" type="checkbox"/>	15	V	
La especificación algebraica de la siguiente operación eliminaría todas las claves repetidas de un determinado ítem (C: ConjuntoConClavesRepetidas; x, y: Ítem):	<input type="checkbox"/>	<input checked="" type="checkbox"/>			
$\text{Eliminar}(\text{Crear}, x) \Leftrightarrow \text{Crear}$ $\text{Eliminar}(\text{Insertar}(C, x), y) \Leftrightarrow$ $\quad \text{si } (x == y) \text{ entonces } C \text{ sino Insertar}(\text{Eliminar}(C, y), x)$	<input type="checkbox"/>	<input checked="" type="checkbox"/>			
En una tabla de dispersión cerrada con la siguiente función de redispersión para la clave 14: $h_i(14)=(28 + 7*i) \text{ MOD } 2000$, se recorrerán todas las posiciones de la tabla buscando una posición libre.	<input type="checkbox"/>	<input checked="" type="checkbox"/>			
El TAD Cola de Prioridad representado por un montículo, tendrá las siguientes complejidades: $O(1)$ para el borrado, y $O(\log n)$ para la inserción, siendo n el número de elementos.	<input type="checkbox"/>	<input checked="" type="checkbox"/>			
En un árbol binario lleno, el camino mínimo de la raíz (longitud del camino más corto hasta un árbol vacío) es igual a la altura del árbol.	<input type="checkbox"/>	<input checked="" type="checkbox"/>			
La altura máxima de un árbol de búsqueda digital es “n+1”, siendo n el número de bits de la clave.	<input type="checkbox"/>	<input checked="" type="checkbox"/>			

Dentro de la especificación algebraica de los números naturales definimos la sintaxis de la función F como: F: natural \rightarrow BOOL, y su semántica como: F(cero)=TRUE, F(suc(cero))=FALSE, F(suc(suc(x)))=F(x). Para el número natural x=35, la función F devolvería FALSE.

En C++, la memoria que se reserva con new se libera automáticamente por el destructor.

La mejor complejidad temporal que se puede conseguir en un algoritmo es O(n), con "n" como la talla del problema.

La semántica de la operación *sublista* del tipo lista vista en clase es la siguiente:

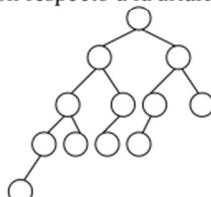
```
VAR L: lista; x, y: item; n: natural; p: posicion;
sublista( L, p, 0 ) = crear()
sublista( crear(), p, n ) = crear()
si p == primera( inscabeza( L, x ) ) entonces
    sublista( inscabeza( L, x ), p, n ) = inscabeza( sublista( L, primera( L ), n ), x )
si no sublista( inscabeza( L, x ), p, n ) = sublista( L, p, n )
```

Dado un único recorrido de un árbol binario lleno, es posible reconstruir dicho árbol

Cuando realizamos un recorrido en inorden en un árbol binario de búsqueda las etiquetas aparecen ordenadas de menor a mayor

Todo árbol binario de búsqueda es un árbol mínimo

El siguiente árbol está balanceado con respecto a la altura



Todo árbol binario de búsqueda es un árbol 2-3.

En un árbol 2-3-4 sólo los nodos hoja y la raíz pueden ser de tipo 2-nodo

En un árbol rojo-negro la búsqueda de una etiqueta dependerá de los colores de los hijos de cada nodo

El árbol 2-3 es un árbol B m-camino de búsqueda con m=2

El TAD Diccionario es un subtipo del TAD Conjunto

En C++, la expresión return *c; devuelve la dirección de memoria de la variable c.

En un multigrafo pueden existir infinitas aristas para un número "n" de vértices.

La semántica de la operación obtener del tipo lista vista en clase es la siguiente:

VAR L1: lista; x: item; p: posicion;

```
obtener( crear(), p ) = error_item()
si p == primera( inscabeza( L1, x ) ) entonces
    obtener( inscabeza( L1, x ), p ) = x
    si no obtener( inscabeza( L1, x ), p ) = inscabeza( obtener( L1, p ), x )
```

El nivel de un nodo en un árbol coincide con la longitud del camino desde la raíz a dicho nodo

A los árboles generales también se les llama árboles multicamino de búsqueda

Un árbol binario de búsqueda equilibrado respecto a la altura tiene una complejidad temporal en su peor caso en la búsqueda de $O(\log_2(n))$, con n el número de elementos del árbol.

En un árbol 2-3 la altura siempre disminuye si al borrar un elemento se produce una combinación con los elementos de la raíz del árbol

En la operación de borrado de un elemento en un árbol 2-3-4, si hay que realizar reestructuraciones, éstas se realizan desde las hojas hacia la raíz.

Las rotaciones en un árbol Rojo – Negro requieren un cambio de color en los nodos implicados.

Todo árbol binario de búsqueda es un árbol B con m=3.

En la dispersión cerrada puede haber colisiones entre claves sinónimas y no sinónimas.

Un Heap Mínimo es un árbol binario que además es árbol mínimo

En un árbol leftist se cumple que:

$CMIN(x) = 1 + CMIN(HijoDer(x))$ para todo x no vacío y x con dos hijos

V	F
<input type="checkbox"/>	<input type="checkbox"/>

1 V

V	F
<input type="checkbox"/>	<input type="checkbox"/>

2 F

V	F
<input type="checkbox"/>	<input type="checkbox"/>

3 F

V	F
<input type="checkbox"/>	<input type="checkbox"/>

4 F

V	F
<input type="checkbox"/>	<input type="checkbox"/>

5 V

V	F
<input type="checkbox"/>	<input type="checkbox"/>

6 V

V	F
<input type="checkbox"/>	<input type="checkbox"/>

7 F

V	F
<input type="checkbox"/>	<input type="checkbox"/>

8 V

V	F
<input type="checkbox"/>	<input type="checkbox"/>

9 F

V	F
<input type="checkbox"/>	<input type="checkbox"/>

10 F

V	F
<input type="checkbox"/>	<input type="checkbox"/>

11 F

V	F
<input type="checkbox"/>	<input type="checkbox"/>

12 F

V	F
<input type="checkbox"/>	<input type="checkbox"/>

13 V

V	F
<input type="checkbox"/>	<input type="checkbox"/>

1 F

V	F
<input type="checkbox"/>	<input type="checkbox"/>

2 V

V	F
<input type="checkbox"/>	<input type="checkbox"/>

3 F

V	F
<input type="checkbox"/>	<input type="checkbox"/>

4 F

V	F
<input type="checkbox"/>	<input type="checkbox"/>

5 F

V	F
<input type="checkbox"/>	<input type="checkbox"/>

6 V

V	F
<input type="checkbox"/>	<input type="checkbox"/>

7 F

V	F
<input type="checkbox"/>	<input type="checkbox"/>

8 F

V	F
<input type="checkbox"/>	<input type="checkbox"/>

9 F

V	F
<input type="checkbox"/>	<input type="checkbox"/>

10 F

V	F
<input type="checkbox"/>	<input type="checkbox"/>

11 V

V	F
<input type="checkbox"/>	<input type="checkbox"/>

12 F

V	F
<input type="checkbox"/>	<input type="checkbox"/>

13 V

V	F		
<input type="checkbox"/>	<input type="checkbox"/>	1	V
<input type="checkbox"/>	<input type="checkbox"/>	2	F
<input type="checkbox"/>	<input type="checkbox"/>	3	F
<input type="checkbox"/>	<input type="checkbox"/>	4	V
<input type="checkbox"/>	<input type="checkbox"/>	5	F
<input type="checkbox"/>	<input type="checkbox"/>	6	F
<input type="checkbox"/>	<input type="checkbox"/>	7	V
<input type="checkbox"/>	<input type="checkbox"/>	8	V
<input type="checkbox"/>	<input type="checkbox"/>	9	F
<input type="checkbox"/>	<input type="checkbox"/>	10	F
V	F		
<input type="checkbox"/>	<input type="checkbox"/>	1	F
<input type="checkbox"/>	<input type="checkbox"/>	2	V
<input type="checkbox"/>	<input type="checkbox"/>	3	F
<input type="checkbox"/>	<input type="checkbox"/>	4	V
<input type="checkbox"/>	<input type="checkbox"/>	5	F
<input type="checkbox"/>	<input type="checkbox"/>	6	F
<input type="checkbox"/>	<input type="checkbox"/>	7	V
<input type="checkbox"/>	<input type="checkbox"/>	8	V
<input type="checkbox"/>	<input type="checkbox"/>	9	F
<input type="checkbox"/>	<input type="checkbox"/>	10	V
<input type="checkbox"/>	<input type="checkbox"/>	11	V
<input type="checkbox"/>	<input type="checkbox"/>	12	F
<input type="checkbox"/>	<input type="checkbox"/>	13	F
<input type="checkbox"/>	<input type="checkbox"/>	14	V

En C++, la expresión `return &c;` devuelve la dirección de memoria de la variable c.

En C++, una función no puede tener todos sus parámetros con valores por omisión o por defecto.

En la escala de complejidades se cumple que $O(\log n) \subset O(\log \log n)$.

La operación BorrarItem, que borra todas las ocurrencias del ítem i que se encuentren en la lista, tiene la siguiente sintaxis y semántica:

```

BorrarItem: LISTA, ITEM -> LISTA
BorrarItem( Crear, i) =      Crear
BorrarItem( IC(L1,j), i) =   si ( i == j ) entonces BorrarItem ( L1, i )
                           sino IC ( BorrarItem ( L1, i ), j )

```

Un árbol con un único nodo tiene un único camino cuya longitud es 1

En cualquier tipo de datos árbol, cada elemento puede tener varios predecesores, pero como máximo un sucesor.

El siguiente árbol está balanceado con respecto a la altura

Si se inserta un elemento en un árbol 2-3 y todos los nodos que están en el camino desde la raíz a la hoja donde se inserta el elemento son del tipo 3-nodo, la altura del árbol 2-3 resultante crece con respecto al árbol 2-3 original.

En un árbol 2-3-4 de altura 3 donde todos sus nodos son del tipo 3-nodo, el número de elementos total es 27.

En un árbol rojo-negro, el número de enlaces negros ha de ser mayor que el de enlaces rojos.

EsVacia: PILA -> BOOLEAN

Si P y Q son pilas: $Q = \text{EsVacia}(P)$, es una expresión sintácticamente correcta

En C++, cuando se sobrecarga un operador que no modifica al operando izquierdo (por ejemplo : "+") se debe crear un objeto temporal, que luego el método devuelve por valor

La complejidad temporal (en su caso promedio) del siguiente fragmento de código es $\Theta(n^2)$

```

int i, length, n, i1, i2, k;
for (i = 0, length = 1; i < n-1; i++) {
    for (i1 = i2 = k = i; k < n-1 && a[k] < a[k+1]; k++, i2++);
    if (length < i2 - i1 + 1) length = i2 - i1 + 1;
}

```

En cualquier tipo de datos lineal cada elemento tiene como máximo un único sucesor y un único predecesor

El máximo número de nodos en un árbol binario de altura $k-1$ es $2^k - 1$, $k \geq 1$.

En la inserción, en el peor de los casos, las rotaciones realizadas en los árboles AVL para mantenerlos balanceados tienen un coste temporal lineal respecto al número de ítems del árbol

El borrado de un elemento en un árbol 2-3 se realiza en las hojas. Se pueden producir reestructuraciones del árbol en el camino de vuelta desde las hojas a la raíz del árbol.

En un árbol 2-3-4 de altura=2 y número de elementos=15, si se insertara un nuevo elemento se tendría que hacer un DIVIDERAIZ y un DIVIDEHIJODE2

Un árbol rojo-negro es un árbol B con $m=2$

Todo árbol binario de búsqueda es un árbol B con $m=4$.

La complejidad temporal en su peor caso de la operación de Unión entre 2 conjuntos con m elementos cada uno y representados con una lista desordenada es $O(m^2)$.

En dos tablas de dispersión cerrada y abierta con tamaños $B=7$ y $B=6$ respectivamente, siempre se cumple que el factor de carga en la abierta es mayor que en la cerrada.

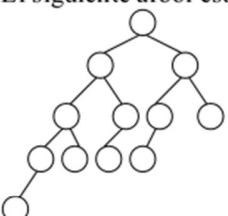
Todo árbol binario que además es árbol mínimo es un Heap Mínimo

La complejidad temporal, en su peor caso, de la operación de PERTENECE de un elemento de tamaño N en un árbol de búsqueda digital es $O(N+1)$.

V	F		
<input type="checkbox"/>	<input type="checkbox"/>	1	V
<input type="checkbox"/>	<input type="checkbox"/>	2	V
<input type="checkbox"/>	<input type="checkbox"/>	3	V
<input type="checkbox"/>	<input type="checkbox"/>	4	F
<input type="checkbox"/>	<input type="checkbox"/>	5	F
<input type="checkbox"/>	<input type="checkbox"/>	6	F
<input type="checkbox"/>	<input type="checkbox"/>	7	F
<input type="checkbox"/>	<input type="checkbox"/>	8	F
<input type="checkbox"/>	<input type="checkbox"/>	9	F
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	10	V

Longitud: LISTA -> NATURAL
Si L es una lista, a es un ítem de la lista: a = Longitud (L) es un uso sintácticamente incorrecto de la operación
En layering los métodos de la clase derivada pueden acceder a la parte pública de la clase base.
En C++, el valor de la variable q al finalizar este fragmento de código es 7:
int q = 0;
int i;
for(i = 1; i < 5; i = i + 1)
 if(i != q)
 q += i;
En la escala de complejidades se cumple que $O(\log n) \subset O(\log \log n)$.
En cualquier tipo de datos lineal cada elemento tiene un único sucesor y varios predecesores
Un árbol binario completo con n nodos y altura k es un árbol binario lleno para esa misma altura
El menor elemento en un árbol binario de búsqueda siempre se encuentra en un nodo hoja
Los árboles AVL son aquellos en los que el número de elementos en los subárboles izquierdo y derecho difieren como mucho en 1
Un árbol 2-3 es un árbol 2-ario de búsqueda
El Árbol 2-3-4 no necesita tener como mínimo una clave en cada nodo

V	F		
<input type="checkbox"/>	<input type="checkbox"/>	1	V
<input type="checkbox"/>	<input type="checkbox"/>	2	F
<input type="checkbox"/>	<input type="checkbox"/>	3	F
<input type="checkbox"/>	<input type="checkbox"/>	4	F
<input type="checkbox"/>	<input type="checkbox"/>	5	V
<input type="checkbox"/>	<input type="checkbox"/>	6	V
<input type="checkbox"/>	<input type="checkbox"/>	7	V
<input type="checkbox"/>	<input type="checkbox"/>	8	F
<input type="checkbox"/>	<input type="checkbox"/>	9	F
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	10	V

En C++, la instrucción "int &a = 1 ; " daría error de compilación
En C++, cuando se emplea layering, desde la clase A que contiene un objeto de la clase B siempre se puede acceder a la parte privada del objeto contenido de la clase B
Paso de programa es una secuencia de operaciones con contenido semántico cuyo coste es dependiente de la talla del problema
La semántica de la operación cima del tipo pila vista en clase es la siguiente:
VAR p: pila, e: ítem;
cima(crear()) = error()
cima(apilar(p, e)) = cima(p)
Dado un único recorrido de un árbol binario lleno, es posible reconstruir dicho árbol
En el borrado de un elemento que se encuentre en un nodo con dos hijos no vacíos en un árbol binario de búsqueda, tenemos que intercambiar el elemento a borrar por el mayor del subárbol de la izquierda o por el menor del subárbol de la derecha
El siguiente árbol está balanceado con respecto a la altura

El borrado de un elemento en un árbol 2-3 se realiza en las hojas. Se pueden producir reestructuraciones del árbol aplicando el algoritmo descendente que empieza en la raíz del árbol y finaliza en las hojas.
En un árbol 2-3-4 el máximo número de elementos del nivel N es $3*2^N-1$
La especificación algebraica de la siguiente operación eliminaría todas las claves repetidas de un determinado ítem (C: ConjuntoConClavesRepetidas; x, y: Ítem):
 Eliminar(Crear, x) \Leftrightarrow Crear
 Eliminar(Insertar(C, x), y) \Leftrightarrow
 si (x == y) entonces C sino Insertar(Eliminar(C, y), x)
En el TAD Diccionario con dispersión cerrada, los elementos se almacenan en una tabla de tamaño fijo.
Todo Heap Mínimo cumple las condiciones de ser un árbol binario y un árbol mínimo
En un multigrafo pueden existir infinitas aristas para un número "n" de vértices.
En un grafo dirigido con K arcos (el número máximo de arcos en el grafo) y N vértices, una complejidad de $O(K)$ es equivalente a la complejidad de $O(N^2)$.

V	F		
<input type="checkbox"/>	<input type="checkbox"/>	1	F
<input type="checkbox"/>	<input type="checkbox"/>	2	V
<input type="checkbox"/>	<input type="checkbox"/>	3	V
<input type="checkbox"/>	<input type="checkbox"/>	4	V
<input type="checkbox"/>	<input type="checkbox"/>	5	F
<input type="checkbox"/>	<input type="checkbox"/>	6	V
<input type="checkbox"/>	<input type="checkbox"/>	7	V
<input type="checkbox"/>	<input type="checkbox"/>	8	F
<input type="checkbox"/>	<input type="checkbox"/>	9	F
<input type="checkbox"/>	<input type="checkbox"/>	10	F
<input type="checkbox"/>	<input type="checkbox"/>	11	V
<input type="checkbox"/>	<input type="checkbox"/>	12	F
<input type="checkbox"/>	<input type="checkbox"/>	13	F
<input type="checkbox"/>	<input type="checkbox"/>	14	F

Longitud: LISTA -> NATURAL
Si L es una lista, a es un ítem de la lista: a = Longitud (L) es un uso sintácticamente correcto de la operación Longitud.

Sea el método Primera perteneciente a la clase Tlista que devuelve la primera posición de la lista que lo invoca:

```
TPosicion Tlista::Primera()
{
    TPosicion p;
    p.pos = lis;
    return p;
}
```

En el método Primera se invoca al constructor y destructor para el objeto TPosicion p.

El algoritmo de intercambio directo o burbuja estudiado en clase (ordenación de los elementos de un vector) tiene una complejidad de $\Omega(n^2)$, siendo n el número de elementos del vector.

La operación BorrarItem, que borra todas las ocurrencias del ítem i que se encuentren en la lista, tiene la siguiente sintaxis y semántica:

```
BorrarItem: LISTA, ITEM -> LISTA
BorrarItem( Crear, i) = Crear
BorrarItem( IC(L1,j), i) = si ( i == j ) entonces BorrarItem ( L1, i )
                           sino IC ( BorrarItem ( L1, i ), j )
```

Existe al menos un árbol binario, que representa los siguientes recorridos: inorder = YXZT, niveles = XTYZ.

El coste temporal (en su peor caso) de insertar una etiqueta en un árbol binario de búsqueda es lineal con la altura del árbol.

Un árbol completo siempre está balanceado respecto a la altura.

El grado del árbol 2-3 es 2.

En un árbol 2-3-4 sólo los nodos hoja y la raíz pueden ser de tipo 2-nodo.

En los conjuntos representados como listas no ordenadas, la complejidad temporal de la operación “diferencia de conjuntos” es $O(n)$, siendo n el número de elementos de cada conjunto.

En la dispersión cerrada puede haber colisiones entre claves sinónimas y no sinónimas.

La definición de un Heap Mínimo indica que ha de ser un árbol binario que además es árbol mínimo.

En un grafo dirigido pueden existir infinitas aristas para un número “n” de vértices.

Sea G un grafo no dirigido de n vértices. Si G tiene “n-1” aristas, entonces nunca podría tener un ciclo.