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.	❑	❑	1	V
Dada la sintaxis de la función *IC(lista,item)* → *lista*, que inserta un elemento a la cabeza de la lista pasada como parámetro y *crear()* → *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*→*b*→*c*, donde *a* es el primer elemento de la lista	❑	❑	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 TRUE.	❑	❑	3	F
En C++, si un objeto se sale de ámbito entonces se invoca automáticamente al destructor de ese objeto.	❑	❑	4	V
En C++, el constructor de copia recibe como argumento un objeto del mismo tipo pasado por referencia o por valor.	❑	❑	5	F
El algoritmo de búsqueda binaria estudiado en clase (búsqueda de un elemento en un vector ordenado) tiene una complejidad de $\Omega(1)$.	❑	❑	6	V
La complejidad espacial es la cantidad de recursos espaciales que un algoritmo consume o necesita para su ejecución	❑	❑	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)	❑	❑	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)	❑	❑	9	F
El máximo número de nodos en un nivel i-1 de un árbol binario es 2^{i-2} , $i \geq 2$	❑	❑	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)	❑	❑	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	❑	❑	12	V
Se puede obtener un único árbol 2-3-4 a partir de su recorrido por niveles	❑	❑	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**		
El algoritmo de búsqueda binaria estudiado en clase (búsqueda de un elemento en un vector ordenado) tiene una complejidad de $\Omega(1)$.	❑	❑	1	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	❑	❑	2	V
Dada la sintaxis de la función *IC(lista,item) → lista*, que inserta un elemento a la cabeza de la lista pasada como parámetro y *crear() → 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 →b →c*, donde *a* es el primer elemento de la lista	❑	❑	3	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 TRUE.	❑	❑	4	F
El máximo número de nodos en un nivel i-1 de un árbol binario es 2^{i-2} , $i \geq 2$	❑	❑	5	V
En C++, el constructor de copia recibe como argumento un objeto del mismo tipo pasado por referencia o por valor.	❑	❑	6	F
En C++, si un objeto se sale de ámbito entonces se invoca automáticamente al destructor de ese objeto.	❑	❑	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)	❑	❑	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)	❑	❑	9	F
La complejidad espacial es la cantidad de recursos espaciales que un algoritmo consume o necesita para su ejecución	❑	❑	10	V
La complejidad logarítmica aparece en algoritmos que descartan muchos valores (generalmente la mitad) en un único paso.	❑	❑	11	V
Se puede obtener un único árbol 2-3-4 a partir de su recorrido por niveles	❑	❑	12	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)	❑	❑	13	F

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→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.	❑	❑	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)	❑	❑	2	V
La complejidad logarítmica aparece en algoritmos que descartan muchos valores (generalmente la mitad) en un único paso.	❑	❑	3	V
La complejidad espacial es la cantidad de recursos espaciales que un algoritmo consume o necesita para su ejecución	❑	❑	4	V
En C++, si un objeto se sale de ámbito entonces se invoca automáticamente al destructor de ese objeto.	❑	❑	5	V
En C++, el constructor de copia recibe como argumento un objeto del mismo tipo pasado por referencia o por valor.	❑	❑	6	F
El máximo número de nodos en un nivel i-1 de un árbol binario es 2^{i-2} , $i \geq 2$	❑	❑	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)	❑	❑	8	F
Se puede obtener un único árbol 2-3-4 a partir de su recorrido por niveles	❑	❑	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	❑	❑	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)	❑	❑	11	F
El algoritmo de búsqueda binaria estudiado en clase (búsqueda de un elemento en un vector ordenado) tiene una complejidad de Ω(1).	❑	❑	12	V
Dada la sintaxis de la función *IC(lista,item)* → *lista*, que inserta un elemento a la cabeza de la lista pasada como parámetro y *crear()*→ *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*→*b*→*c*, donde *a* es el primer elemento de la lista	❑	❑	13	F