Apellidos, Nombre:
DNI:

Examen PED abril 2016
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, una operación es una función que toma como parámetros (entrada) uno o más valores de diversos tipos, y produce como resultado un solo valor de otro tipo.	❑	❑	1	F
Las ecuaciones (vistas en clase) que permiten realizar la multiplicación de números naturales son las siguientes: VAR x, y: natural; mult(cero, x) = cero mult(x, cero) = cero mult(suc(y), x) = suma(mult(y, x), x)	❑	❑	2	V
En la especificación algebraica, para el tratamiento de errores se añade una constate a la signatura que modeliza un valor de error, por ejemplo $error_{nat}$ → natural.	❑	❑	3	V
En C++, si se declara un objeto *a* (p. ej. *TPoro a;*) cuando la variable *a* se sale de ámbito entonces se invoca automáticamente al destructor de ese objeto.	❑	❑	4	V
Las ecuaciones (vistas en clase) para la operación *recu* de un vector son las siguientes: recu(crear( ), i) = error( ) recu(asig(v, i, x), j) **si** (i == j) **entonces** j **si no** recu(v, j) **fsi**	❑	❑	5	F
La complejidad temporal de la operación *desapilar* (vista en clase) utilizando vectores (con un índice que indica la cima de la pila) o utilizando listas enlazadas es la misma.	❑	❑	6	V

La complejidad temporal del siguiente fragmento de código es $O(n^2)$
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; }
	❑	❑	7	F

En las colas circulares enlazadas vistas en clase, las operaciones *encolar* y *desencolar* tienen complejidad temporal $\Theta(1)$.	❑	❑	8	V
Las ecuaciones (vistas en clase) para la operación *desencolar* son las siguientes: desencolar(crear( )) = crear( ) **si** esvacia(c) **entonces** desencolar(encolar(c, x)) = crear ( ) **si no** desencolar(encolar(c, x)) = encolar(desencolar (c), x)	❑	❑	9	V
Es posible reconstruir un único árbol binario de búsqueda a partir de un recorrido en preorden.	❑	❑	10	V
Un camino en un árbol es una secuencia $a_1, ..., a_s$ de árboles tal que para todo $i \in \{1, ..., s-1\}$, a_i es subárbol de a_{i+1}.	❑	❑	11	F
A los árboles generales también se les llama árboles multicamino de búsqueda.	❑	❑	12	F
La semántica de la operación *quita_hojas* que actúa sobre un árbol binario y devuelve el árbol binario original sin sus hojas es la siguiente: VAR i, d: arbin; x: item; quita_hojas(crea_arbin( )) = crea_arbin( ) quita_hojas(enraizar(crea_arbin(), x, crea_arbin()) = enraizar(crea_arbin(), x, crea_arbin() quita_hojas(enraizar(i, x, d)) = enraizar(quita_hojas(i), x, quita_hojas(d))	❑	❑	13	F
Profundidad de un subárbol es la longitud del único camino desde la raíz a dicho subárbol.	❑	❑	14	V