Apellidos:
Nombre:
Convocatoria:
DNI:

Examen PED junio 2007
Modalidad 0

Normas:
- La entrega del test no corre convocatoria.
- 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		
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*.	❑	❑	1	F
Restricciones al sobrecargar los operadores en C++: no es posible modificar el comportamiento del operador "+" con tipos predefinidos del lenguaje (por ejemplo el tipo *int*)	❑	❑	2	V
En C++, el valor de la variable *q* al finalizar este fragmento de código es 16: *int q = 1;* *while(q < 12)* *q *= 2;*	❑	❑	3	V
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.	❑	❑	4	V
La semántica de la operación *cima* del tipo pila vista en clase es la siguiente: *VAR p: pila, e: item;* *cima(crear( )) = error( )* *cima(apilar(p, e)) = cima(p)*	❑	❑	5	F
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 mínimo un sucesor.	❑	❑	6	F
El siguiente vector representa un montículo máximo: 10 | 5 | 3 | 1 | 2	❑	❑	7	V
El número máximo de arcos de un grafo dirigido es mayor que el número de aristas de un grafo no dirigido con el mismo número de vértices.	❑	❑	8	V
En un diccionario implementado como un trie, la operación de búsqueda tendrá una complejidad temporal de *O(L)* siendo *L* la longitud de la cadena buscada.	❑	❑	9	V
Para todo nodo de un árbol Leftist mínimo, se cumple que las claves de los hijos son menores que la del padre.	❑	❑	10	F
Cuando utilizamos una tabla de dispersión cerrada de tamaño B, el número máximo de elementos que se pueden almacenar está limitado a B elementos.	❑	❑	11	V
Todo árbol B con m=4 cumple las condiciones exigidas para ser un árbol 2-3-4.	❑	❑	12	V

Examen PED junio 2007

Normas:
- Tiempo para efectuar el ejercicio: **2 horas**
- En la cabecera de cada hoja **Y EN ESTE ORDEN** hay que poner: ***Apellidos, Nombre***.
- Cada pregunta se escribirá en hojas diferentes.
- Se dispone de 20 minutos para abandonar el examen sin que corra convocatoria.
- Las soluciones al examen se dejarán en el campus virtual.
- Se puede escribir el examen con lápiz, siempre que sea legible
- **Todas las preguntas tienen el mismo valor.** Este examen vale el 60% de la nota de teoría.
- **Publicación de notas de exámenes de teoría**: 2 de julio. **Fecha de revisión de prácticas:** 3 de julio **16:00 h.** en aula LS13i de la EPS IV
  Fecha de revisión de examen de teoría: 4 de julio, 9:00h. en aula LS13i de la EPS IV
- **Los alumnos que estén en 5ª o 6ª convocatoria deben indicarlo en la cabecera de todas las hojas**

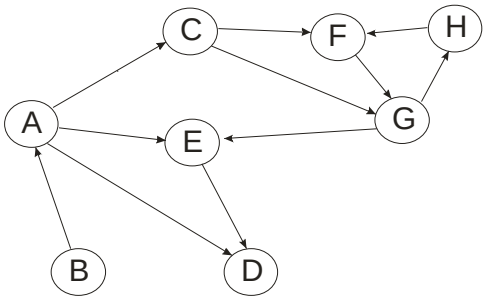
1. Utilizando las operaciones definidas en clase para la definición del tipo *vector* definir la sintaxis y la semántica de la operación *palindromo* que indica si un vector de naturales con 100 elementos es palíndromo. Por ejemplo, el vector 1,25,12,3,3,12,25,1 es palíndromo (se ha simplificado el ejemplo con un vector de 8 elementos). IMPORTANTE: se asume que el vector está creado con tamaño 100, está lleno y el rango de las posiciones es de 1 a 100. Se premiará la solución temporalmente más eficiente.

2. En un árbol AVL inicialmente vacío, realiza sucesivamente las siguientes operaciones (IMPORTANTE: indica en cada momento el factor de equilibrio de cada nodo y el tipo de transformación necesaria para reequilibrar el árbol. Criterio: en caso de dos hijos, elige el menor de la derecha):

a) Inserta las etiquetas 30, 40, 50, 20, 10, 35, 55
b) Borra las etiquetas 10, 40, 55
c) Inserta las etiquetas 70, 32
d) Borra las etiquetas 30, 32, 20

3. Sobre el siguiente grafo dirigido (IMPORTANTE: la lista de adyacencia de cada nodo está ordenada alfabéticamente de menor a mayor):

a) Realiza el recorrido DFS(A) y obtén el bosque extendido en profundidad.
b) Etiqueta los arcos.
c) Realiza el recorrido BFS(A).



4. **a)** Aplicar el algoritmo *HeapSort* para ordenar el siguiente vector **de mayor a menor** (sin utilizar un vector auxiliar). Mostrar en cada iteración del algoritmo, el montículo representado gráficamente y dentro del propio vector.

b) Indicar justificadamente la complejidad temporal de este algoritmo.

10	15	2	6	20	30	5	9	40	1

Examen PED junio 2007. Soluciones

1.

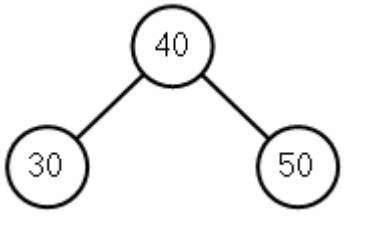
Sintaxis
palindromo: vector → bool

Semántica
Var v: vector; i,x: natural;

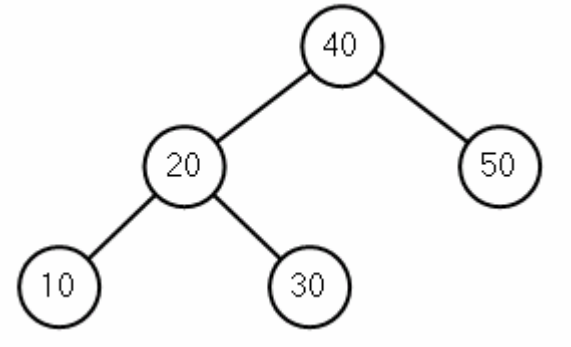
palindromo(crear_vector()) = VERDADERO
palindromo(asig(v,i,x)) = si i<= 50
entonces si recu(v,100-i+1) == x
entonces palindromo(v)
sino FALSO
sino palindromo(v)

2.

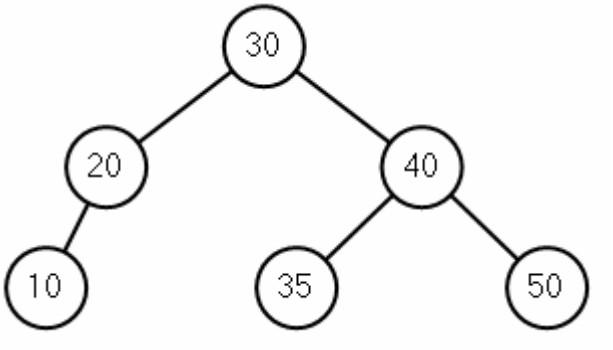
Apartado a)
Inserta 30, 40, 50 --> Rotación DD



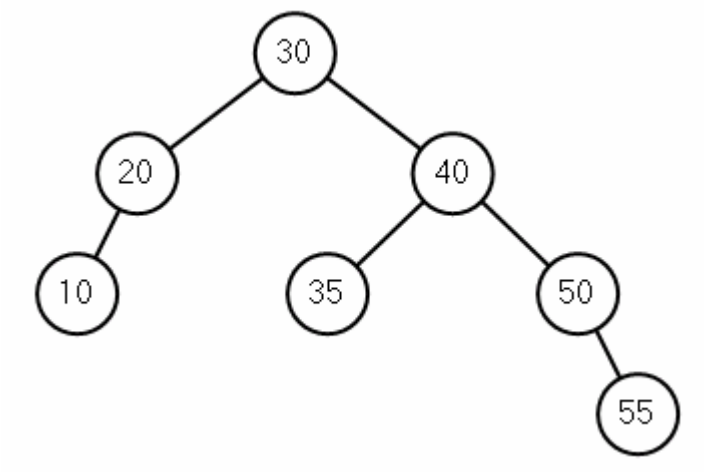
Inserta 20, 10 --> Rotación II



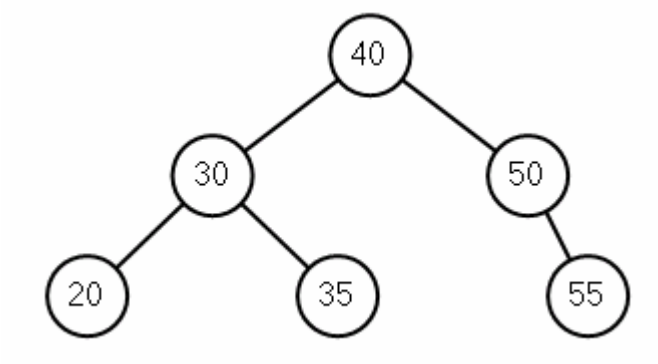
Inserta 35 --> Rotación ID



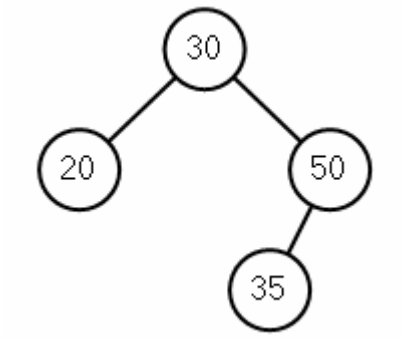
Inserta 55



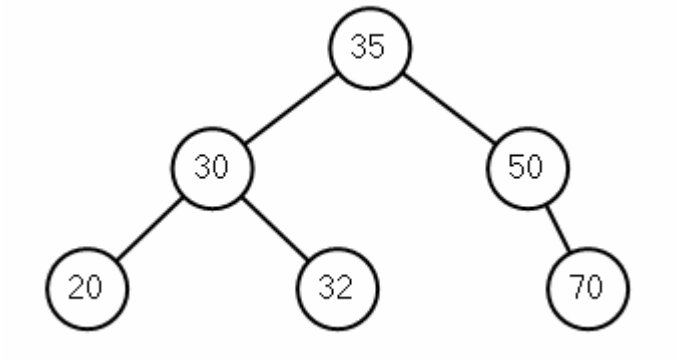
Apartado b)
Borra 10 --> Rotación DD



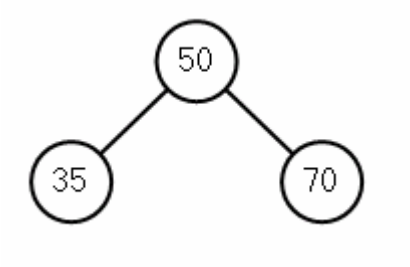
Borra 40, 55 --> Rotación II



Apartado c)
Inserta 70, 32 --> Rotación DI

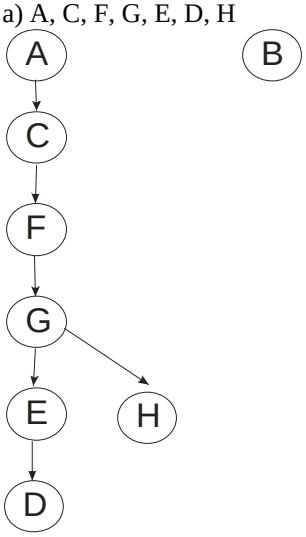


Apartado d)
Borra 30, 32, 20 --> Rotación DD



3.

a) A, C, F, G, E, D, H



b)
(A, C) = Arbol.
(C, F) = Arbol.
(F, G) = Arbol
(G, E) = Arbol
(G, H) = Arbol
(E, D) = Arbol
(A, E) = Avance.
(A, D) = Avance.
(C, G) = Avance.
(H, F) = Retroceso.
(B, A) = Cruce.

c) A,C,D,E,F,G,H

4. a) El algoritmo se aplicaría del siguiente modo:

1. Dejamos la parte izquierda del vector para obtener un montículo **mínimo** donde insertar todos los elementos del vector. La parte derecha se utilizará para almacenar los elementos todavía no insertados.
2. Cuando el montículo esté lleno, repetidamente se borra la raíz del montículo llevándola a la parte derecha del vector.
3. Al quedar vacío el montículo, el vector nos quedará ordenado de mayor a menor.

40	30	20	15	10	9	6	5	2	1

b) Con *n* el número de elementos del vector, se hacen *2n* operaciones de insertar y borrar, cada una de ellas con coste logarítmico **(**dado que se realizan sobre un montículo, en el que estas operaciones tienen una complejidad lineal en función de la altura del árbol completo, por lo que es logarítmica en función del número de elementos del montículo), con lo que se concluye que tendrá un coste $O(n \log_2 n)$.