Apellidos:
Nombre:
Convocatoria:
DNI:

Examen PED julio 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.
- Este test vale 4 puntos (sobre 10).
- **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 temporal (en su caso mejor) del siguiente fragmento de código es $\Omega(n)$
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; }
	❑	❑	1	V

La complejidad temporal (en su peor caso) de la operación de insertar un elemento en una cola circular enlazada que no admite elementos repetidos es O(n), siendo n el número de elementos de la cola.	❑	❑	2	V
Un árbol con un único nodo es un árbol completo.	❑	❑	3	V
El nivel de la raíz en un árbol binario es 0.	❑	❑	4	F
Todo árbol binario mínimo es un árbol binario de búsqueda.	❑	❑	5	F
Un árbol binario de búsqueda completo es un AVL.	❑	❑	6	V
El número de rotaciones que se nos pueden dar en el borrado de un elemento en un AVL son como máximo 3 menos que la altura del árbol.	❑	❑	7	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 en el mencionado árbol es $O(\log_2 n)$.	❑	❑	8	V
En un árbol 2-3-4 los nodos pueden tener 1, 2, 3 ó 4 hijos.	❑	❑	9	F
La mejor representación de los conjuntos siempre es el vector de bits porque es la más eficiente espacialmente.	❑	❑	10	F
Sea una tabla de dispersión cerrada con estrategia de redispersión $h_i(x)=(H(x) + C*i) \text{ MOD } B$, con B=1000 y C=74. Para cualquier clave “x” que se desee insertar, se recorrerán todas las posiciones de la tabla buscando una posición libre.	❑	❑	11	F
El siguiente vector representa un montículo máximo: | 10 | 5 | 3 | 1 | 2 |	❑	❑	12	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” $\Rightarrow$ gradoE(v) $\leq$ 1	❑	❑	13	V
Un digrafo es un multigrafo que no contiene arcos reflexivos.	❑	❑	14	F
La especificación algebraica de la operación *longitud* definida en clase para el tipo lista es la siguiente: VAR L1: lista; x: item; *longitud*(crear( )) = 0 *longitud* (inscabeza(L1, x)) = 1 + inscabeza(*longitud* (L1), x)	❑	❑	15	F
En la especificación algebraica de un tipo de datos las operaciones modificadoras devuelven un valor de un tipo diferente al que se está definiendo.	❑	❑	16	F

Examen PED julio 2016

Normas:
- Tiempo para efectuar el examen: **2 horas**
- En la cabecera de cada hoja **Y EN ESTE ORDEN** hay que poner: ***APELLIDOS, NOMBRE***.
- Cada pregunta se escribirá en hojas diferentes.
- Las soluciones al examen se dejarán en el campus virtual.
- Se puede escribir el examen con lápiz, siempre que sea legible
- **Cada pregunta vale 2 puntos (sobre 10).**
- **Las fechas de "Publicación de notas" y "Revisión del examen teórico" se publicarán en el Campus Virtual.**

1. **a)** Utilizando exclusivamente las operaciones constructoras generadoras del **tipo pila**, definir la sintaxis y la semántica de la operación QuitaPares que actúa sobre una pila y devuelve una pila en la que se han eliminado las posiciones pares de la misma.

Nota: se asume que la posición 1 (impar) de la pila está en la cima de la misma.

b) Explicar las dos representaciones enlazadas del **tipo cola** vistas en clase definiendo los elementos que aparecen en la misma. Para cada una de ellas, explicar razonadamente (justificando la respuesta) la complejidad temporal (en su mejor y peor caso) de las operaciones encolar y desencolar.

2.

a) Inserta en un árbol 2-3 inicialmente vacío los siguientes elementos: 10, 25, 20, 50, 60, 15, 30, 80

b) ¿El árbol resultado obtenido en el apartado a) cumple las propiedades de árbol 2-3-4? Justifica tu respuesta. En caso afirmativo, suponiendo que el árbol resultado del apartado a) sea un 2-3-4, inserta los siguientes elementos: 35, 40, 45. En caso negativo, será un 2-3 y se insertarán los siguientes elementos: 70, 75, 5

c) Sobre el resultado del apartado b) borra los elementos 45, 25 e indica de qué tipo es el árbol resultado. Criterios: Si el nodo tiene dos hijos substituir por el mayor de la izquierda. Si se realiza el borrado sobre un 2-3-4, en caso de tener dos nodos adyacentes a q entonces r será el hermano de la derecha. Si se realiza el borrado sobre un 2-3, en caso de tener dos hermanos consultar el hermano de la derecha.

3.

a) Escribir en C++ el código de una función para ordenar un vector de enteros (TVectorEnteros) en orden ascendente o descendente mediante un montículo doble (TDeap). Ejemplo de uso:

TVectorEnteros a; Ordenar(a, true); // Ordenación de menor a mayor

NOTAS: los errores de sintaxis de C++ se puntuarán de forma negativa; no hace falta definir el código de los métodos de las clases TDeap y TVectorEnteros, pero sí que habrá que definir los prototipos de los métodos que se utilicen (parámetros de entrada y salida); no es necesario que se hagan todas las operaciones sobre el mismo vector a ordenar.

b) Indicar RAZONADAMENTE su complejidad en el caso peor.

c) Ordenar el siguiente vector de menor a mayor usando dicha función, explicando las operaciones realizadas:

20	10	30	15	25	40	45	5	3	47	27	32	2	7	50	1

Examen PED julio 2016. Soluciones

1. a) QuitaPares: pila → pila

Var p: pila; x,y: item;

QuitaPares (crear_pila())= crear_pila()
QuitaPares (apilar(crear_pila(), x))= apilar(crear_pila(), x)
QuitaPares (apilar(apilar(p, x), y))= apilar(QuitaPares(p), y)

b) Para la representación enlazada de las colas se utilizan punteros a **nodo**. El nodo contiene el **dato** a almacenar y un **puntero al siguiente nodo**. Se definen dos punteros adicionales: **tope** y **fondo**. **tope** apunta al primer elemento que hay que desencolar y **fondo** apunta al último elemento de la cola.

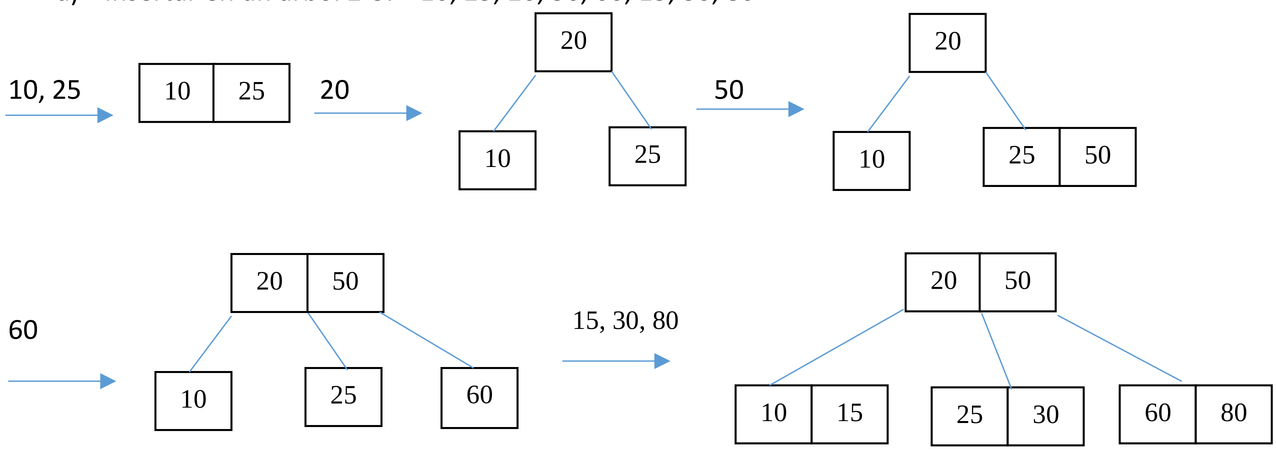
Utilizando esta estructura la complejidad temporal de las operaciones **encolar y desencolar** es la misma (ya que tenemos punteros a la primera y a la última posición de la cola): $\mathbf{\Omega(1) = O(1) = \theta(1)}$

Esta representación se optimiza con las **colas circulares enlazadas**, en las que sólo se necesita un puntero (**fondo**) al último elemento de la cola; el siguiente elemento de fondo apunta al primer elemento de la cola.

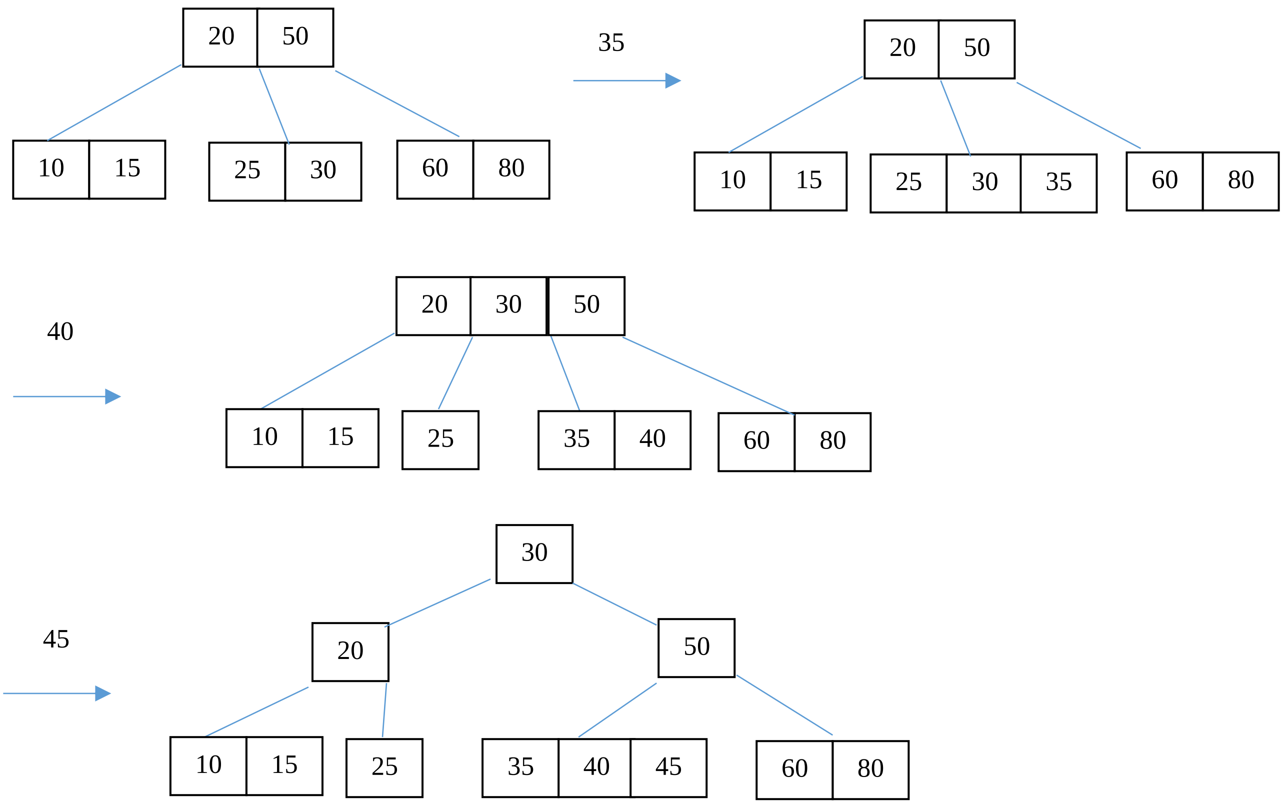
La complejidad temporal de las operaciones **encolar y desencolar** es la misma (ya que tenemos un puntero que apunta a la última posición y al primero –siguiente de fondo–): $\mathbf{\Omega(1) = O(1) = \theta(1)}$

2.

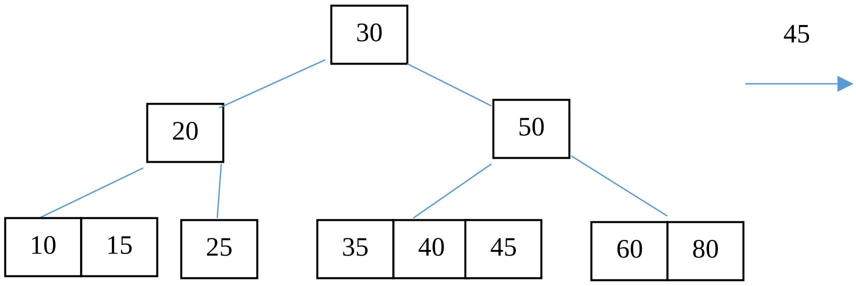
a) Insertar en un árbol 2-3: 10, 25, 20, 50, 60, 15, 30, 80

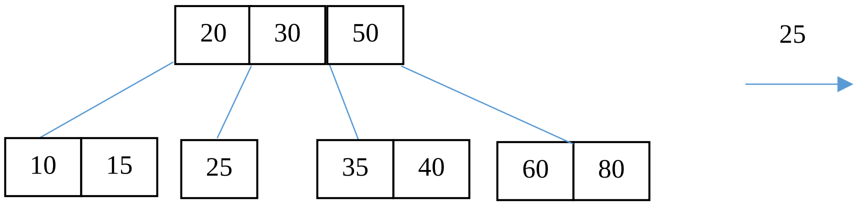


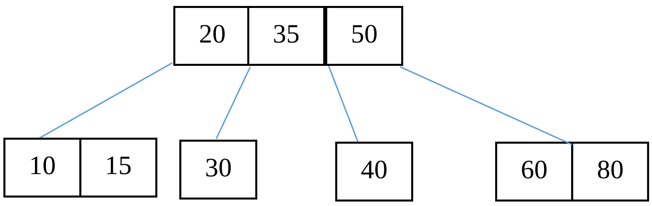
b) Insertar en un árbol 2-3-4: 35, 40, 45



c) Borrar en un árbol 2-3-4: 45,25







3.

a)

```
class TDeap {
	public:
		void Insertar(int);		// Inserta un elemento
		int Maximo(void);		// Devuelve el máximo del Deap
		int Minimo(void);		// Devuelve el mínimo del Deap
		void BorrarMaximo(void);// Borra el máximo del Deap
		void BorrarMinimo(void);// Borra el mínimo del Deap
		…
};
class TVectorEnteros {
	public:
		int Tamanyo(void);		// Devuelve el número de elementos del vector
		…
};

void Ordenar(TVectorEnteros& a, const bool& menorAmayor) {
	TDeap d; int i;

	for(int i  = 1; i <= d.Tamanyo(); ++i) {
		d.Insertar(a[i]);
	}
	if(menorAmayor) {
		for(i  = 1; i <= d.Tamanyo(); ++i) {
			a[i] = d.Minimo();
			d. BorrarMinimo();
		}
	}
```

```
        else {
            for(i  = 1; i <= d.Tamanyo(); ++i) {
                a[i] = d.Maximo();
                d. BorrarMaximo();
            }
        }
    }
```

b) Tendría una complejidad O(n log n), con n el número de elementos del vector, al igual que el algoritmo HeapSort que utiliza un montículo simple, ya que igualmente habría que realizar n operaciones de inserción y borrado, cada una de ellas con una complejidad O(log n).