

<b>APELLIDOS:</b> <b>GRUPO</b> <b>Duración 2h 30m</b>	<b>NOMBRE:</b>
---	----------------

<b>Ejercicio 1: Test</b>	<b>Ejercicio 2</b>	<b>Ejercicio 3</b>	<b>TOTAL</b>

**Ejercicio 1:** [30 puntos: respuesta acertada = +2, respuesta incorrecta = -1]

**Restricción de evaluación: Mínimo test: 12 puntos.**

Conteste a las siguientes cuestiones. Sólo existe UNA respuesta correcta por pregunta

- 1.1. De las siguientes afirmaciones sobre bolsas, señale la que considere falsa.
  - (a) No está definida una relación de orden..
  - (b) Los elementos pueden ser de distinto tipo.
  - (c) Se permiten las repeticiones.
  - (d) Puede implementarse mediante un vector de booleanos.
  
- 1.2. ¿Qué estrategia es la más apropiada para evitar el *clustering* en tablas *Hash*?
  - (a) La prueba lineal.
  - (b) La prueba cuadrática.
  - (c) El encadenamiento separado.
  - (d) Ninguna de las anteriores.
  
- 1.3. En la especificación del TAD Pila, la operación  
**PARCIAL** Desapilar: `TipoPila → TipoPila`
  - (a) Es Constructora Generadora junto a la operación `CrearPilaVacía`.
  - (b) Es Constructora No Generadora porque devuelve el mismo tipo que el definido en el TAD.
  - (c) Es Observadora Selectora porque devuelve una parte de la pila original.
  - (d) Es imposible catalogarla y dependerá de la implementación.
  
- 1.4. ¿Por dónde se insertan los elementos en una Bicola?
  - (a) Por la izquierda.
  - (b) Por la derecha.
  - (c) Por el centro
  - (d) Por la izquierda y por la derecha.
  
- 1.5. La operación `Buscar` en una Lista Enlazada Ordenada tiene una complejidad  $O()$ 
  - (a) Mayor que la de la lista enlazada sin ordenar.
  - (b) Depende de si tiene inserción al final.
  - (c) Igual que la de la lista enlazada sin ordenar.
  - (d) Menor que la de la lista enlazada sin ordenar.

- 1.6. Marque la opción correcta sobre grafos:
- Un grafo no dirigido conexo es fuertemente conexo.
  - Un grafo dirigido conexo es fuertemente conexo.
  - Un grafo conexo es siempre fuertemente conexo independientemente de si es dirigido o no dirigido.
  - Todas son falsas.
- 1.7. Una representación de un grafo mediante una matriz de adyacencia:
- La consulta de la adyacencia de 2 nodos concretos es más eficiente que mediante una representación con una lista de adyacencia.
  - Se utiliza típicamente para grafos dispersos.
  - No se puede utilizar si el grafo es dirigido.
  - Todas las respuestas son correctas.
- 1.8. **[Puntúa doble: +4 / -2]** Dada la siguiente especificación, deducir razonadamente de qué TAD identifica la especificación misteriosa.

---

**ESPECIFICACION** Misteriosa

**TIPOS** TipoEstructura

**PARAMETROS GENERICOS**

**TIPOS** TipoElemento, TipoCaracteristica

**OPERACIONES**

Caracteristica: TipoElemento -> TipoCaracteristica

(\* Caracteristica devuelve la característica especial asociada a un elemento \*)

Mayor: TipoCaracteristica x TipoCaracteristica -> Booleano

(\* Operación de orden total ente características. Dice si la primera es mayor que la segunda o no\*)

**FIN PARAMETROS GENERICOS**

**OPERACIONES**

**(\* CONSTRUCTORAS GENERADORAS \*)**

CrearEstructuraVacia: -> TipoEstructura

Insertar: TipoElemento x TipoEstructura -> TipoEstructura

**(\* OTRAS OPERACIONES \*)**

PARCIAL Misterio: TipoEstructura -> TipoElemento

EsEstructuraVacia: TipoEstructura -> Booleano

**VARIABLES**

e, e1, e2: TipoElemento

estructura: TipoEstructura

**ECUACIONES DE DEFINITUD**

DEF (Misterio (Insertar (e, estructura)))

**ECUACIONES ENTRE GENERADORAS**

Insertar (e1, Insertar (e2, estructura)) = Insertar (e2, Insertar (e1, estructura))

**ECUACIONES**

EsEstructuraVacia (CrearEstructuraVacia) = **CIERTO**

EsEstructuraVacia (Insertar (e, estructura)) = **FALSO**

Misterio (Insertar (e, estructura)) =

SI EsEstructuraVacia(estructura) O Mayor(Caracteristica(e),  
Caracteristica(Misterio(estructura))

-> e

| Misterio (estructura)

**FIN ESPECIFICACION**

---

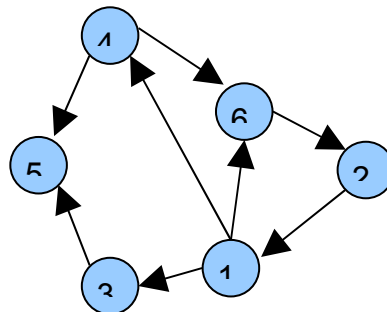
**Respuesta**

- 1.9. **[Puntúa doble: +4 / -2]** Dada la siguiente secuencia de datos y la función *Hash*  $H(k) = (k \text{ MOD } 100) \text{ DIV } 7$ , construir una tabla *Hash* con 15 filas [0..14] donde la clave será el DNI y se resuelvan las colisiones mediante prueba cuadrática.

Secuencia	Nombre	Apellido	DNI
1	Alfonso	Fernández	35.875.690
2	Abraham	Duarte	41.002.101
3	Juanjo	Pantrigo	8.424.235
4	Mayte	González	25.061.994
5	Rafael	Nadal	17.890.335
6	Fernando	Alonso	44.111.692
7	David	Villa	85.669.067
8	Mariano	Rajoy	4.511.020
9	Jose Luis	Rodríguez	75.201.873
10	Julio	Anguita	644.770

**Respuesta:**

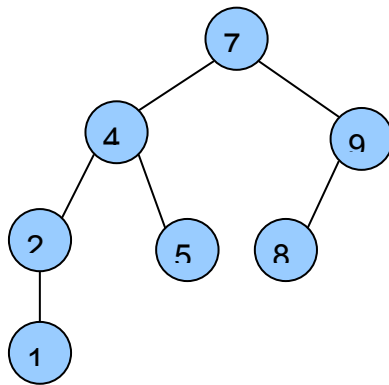
- 1.10. **[Puntúa doble: +4 / -2]** Dado el siguiente grafo y suponiendo que se empieza desde el vértice 2, mostrar el recorrido en profundidad y anchura, así como sus correspondientes árboles asociados,.



**Respuesta:** Recorrido en profundidad:

**Respuesta:** Recorrido en anchura:

1.11. [Puntúa doble: +4 / -2] Dado el árbol AVL que aparece en la figura, representar gráficamente dicho árbol después de eliminar el nodo cuya clave es 7.



**Respuesta:**

**Ejercicio 2: [40 puntos]**

Se desea desarrollar un subprograma que reciba dos colas  $C1$  y  $C2$ , y devuelva como resultado una nueva cola  $C3$ , la cual estará formada por la fusión de las otras dos. Tanto la cola  $C1$  como la  $C2$  pueden tener elementos comunes (entre ambas) y repetidos (dentro de cada una), pero después de la fusión, la cola  $C3$  sólo contendrá aquéllos elementos que sean diferentes.

El método de fusión consistirá en coger alternativamente elementos de la cola  $C1$  y después de la  $C2$ . En el caso de que el elemento a introducir esté ya dentro de la cola  $C3$ , éste se descartará y se probará con el siguiente hasta que se encuentre un elemento no introducido previamente. El proceso de fusión terminará cuando una de las dos colas se acabe.

Para resolver el problema, se deberá utilizar el TAD **ColaEst**, implementado mediante el uso de memoria estática pero *simulando el uso de memoria dinámica* con cursores cuyas operaciones sean de la menor complejidad posible.

Se pide:

- a) Especificación algebraica de **ColaEst**, considerando las operaciones constructoras generadoras así como `Primero`, `EsColaVacía` y `Eliminar`. **[10 puntos]**.
- b) Implementación en Pascal del TAD **ColaEst** apoyándose en la especificación descrita en el apartado anterior. Indicar la complejidad de cada función y procedimiento. **[20 puntos]**
- c) Desarrollar un subprograma que reciba dos colas de tipo **ColaEst** y devuelva como una cola que almacene la fusión descrita previamente. **[10 puntos]**.

**NOTA1:** La operación de recorrido en una cola no está definida en su especificación, por tanto no debe usarse.

**NOTA2:** En el caso de necesitar TADs auxiliares sólo se deberá proporcionar la interfaz, incluyendo una breve descripción de las funciones y procedimientos.







**Ejercicio 3: [30 puntos]**

Considérese que se dispone de un TAD `TString`, donde cada variable de este tipo es una cadena de caracteres de longitud arbitraria. Se puede suponer que se dispone de la siguiente funcionalidad.

```
PROCEDURE Sacapalabra(VAR textInOut:TString; sep:Tseparador; VAR palabra:TString);  
{Recibe un texto textInOut y un separador sep y devuelve en palabra el texto desde  
el origen al primer separador y el resto en textInOut}  
  
FUNCTION EstextoVacio(texto: TString): Boolean;  
{Comprueba si el texto es vacío o no}
```

Se pide:

- a) Implementar un subprograma que reciba un texto de tipo `TString` y que produzca como resultado la lista de todas las palabras diferentes contenidas en el texto así como su frecuencia de aparición. **[10 puntos]**
- b) Implementar un subprograma que reciba la lista descrita en el apartado anterior y genere un árbol binario de búsqueda donde el criterio de ordenación sea la frecuencia de repetición de cada palabra. **[20 puntos]**









