ESTRUCTURA DE DATOS Y ALGORITMOS **6-Septiembre-2010**
Titulación: Ingeniero Técnico en Informática de Gestión **Curso: 2º**
Nombre y apellidos: _____ Nota:

- Escribe tu **nombre y apellidos** en esta hoja e inmediatamente en todas las suplementarias, incluso las de sucio. El no hacerlo puede suponer tu expulsión
- Puedes utilizar el **lápiz** para tus respuestas. No está permitido el uso de apuntes, notas o libros. No puedes tener un **móvil** encendido, ni utilizar cualquier otro **aparato electrónico**.
- **Todos los alumnos implicados en una copia** de un ejercicio **tendrán una nota final de 0**. El alumno es responsable de velar por su examen. Es decir **tanto el que copia como el que se deja copiar (ya sea de manera activa o pasiva) recibirán el mismo castigo sin que exista atenuante alguno**

1. Ejercicio (3 puntos)

a) Relaciona la lista de descripciones al menos con uno de los nombres de las estructuras de datos que están a la derecha.

1. Restricción de operaciones sobre una estructura de datos	( ) Arrays
2. Organización de datos no lineal	( ) Vectores
3. En su definición el uso de la memoria es estática, pero puede variar en la ejecución de un programa	( ) Pilas
	( ) Árboles
4. Uso ineficiente de la memoria	( ) Lista
5. La inserción de un nuevo dato requiere analizar varias casos	

b) En qué estructura de datos se utiliza el término *colisión*? A qué se refiere exactamente este término?

c) La estructura de datos abstracta Pila se conoce también como una estructura ……. Explica el porqué.

d) La solución de un problema concreto ejecuta, con bastante frecuencia, las dos siguientes operaciones sobre un conjunto de datos: 1) buscar un dato por contenido y 2) insertar o eliminar más datos después o antes del dato buscado en el 1). ¿Qué estructura de datos sería la más adecuada? Argumenta tu decisión.

e) Cual de las siguientes afirmaciones es cierta:
   1. Las estructuras de datos almacenan operaciones y las aplican sobre un objeto
   2. Las estructuras de datos restringen el uso de las variables
   3. Las estructuras de datos almacenan variables de objetos
   4. Las estructuras de datos almacenan referencias a objetos

f) Las principales diferencias entre las distintas estructuras son:
   1. La forma de organizar las operaciones y las restricciones que imponen sobre los datos
   2. La forma de organizar los datos y las restricciones que imponen sobre las operaciones
   3. La forma en que aplican la abstracción
   4. En principio, no existe ninguna diferencia sustancial, depende del problema planteado

g) Las tablas hash son utilizadas porque:
   1. Se deben de recorrer los elementos en orden
   2. No se conoce el número de elementos a priori
   3. Se quiere utilizar un mismo índice para varios elementos
   4. Ninguna de las anteriores

h) Un vector tiene la siguiente propiedad:
   1. Inserciones y borrados eficientes en medio de la estructura
   2. Inserciones y borrados eficientes en ambos extremos de la estructura
   3. Inserciones y borrados eficientes al final de la estructura
   4. Ninguna de las anteriores

ESTRUCTURA DE DATOS Y ALGORITMOS **6-Septiembre-2010**
Titulación: Ingeniero Técnico en Informática de Gestión **Curso: 2º**
Nombre y apellidos: _____

- Escribe tu **nombre y apellidos** en esta hoja e inmediatamente en todas las suplementarias, incluso las de sucio. El no hacerlo puede suponer tu expulsión
- Puedes utilizar el **lápiz** para tus respuestas. No está permitido el uso de apuntes, notas o libros. No puedes tener un **móvil** encendido, ni utilizar cualquier otro **aparato electrónico**.
- **Todos los alumnos implicados en una copia** de un ejercicio **tendrán una nota final de 0**. El alumno es responsable de velar por su examen. Es decir **tanto el que copia como el que se deja copiar (ya sea de manera activa o pasiva) recibirán el mismo castigo sin que exista atenuante alguno**

2. *Ejercicio (2,5 puntos)*

Se quiere desarrollar un programa para la visualización de todos los valores de una Bolsa. Los datos que se visualizan por cada valor son: el *nombre*, el *precio* actual de una acción, el *porcentaje* de diferencia respecto al precio de cierre del día anterior y el *volumen* negociado durante el día (y podrían ser más datos). La lista de valores se puede ordenar por cualquier atributo y orden, según la selección realizada por el usuario.

El funcionamiento del programa diariamente es el siguiente: 1) Carga todos los datos del cierre del día anterior, que están almacenados en un fichero de texto (la 1ª vez estará vacía) en el PC, se cargan a una estructura de datos en memoria; 2) Cada 5s. a) se conecta a Internet y en un fichero de texto, recibe la lista de los valores (ordenados por el nombre) cuyos datos han cambiado desde la última conexión (la 1ª vez recibirá todas); b) los datos son actualizados; c) si ha habido algún cambio se vuelve a visualizar la lista de valores según el último criterio de ordenación seleccionado por el usuario; y 3) al final del día realiza el cierre guardando el fichero en el PC. Algunas veces, el usuario puede cambiar el criterio de ordenación seleccionando el nuevo atributo y el orden (ascendente o descendente) por el cual quiere ordenar.

Se dispone de las siguientes clases:

```
public class VerBolsa{
        /**
          * Crea una visualización de Bolsa vacía
          */
        public Verbolsa();
        /**
          *  Carga desde un fichero de texto los datos de los valores según el cierre del día anterior
          */
        public void cargarDatos(String fichero);
        /**
          *  Actualiza los datos de los valores según el fichero recibido con los nuevos datos
          */
        public void actualizar(String fichero);
        /**
          *  Inserta el valor en la lista según el criterio de ordenación
          */
        public void insertarValorOrdenado(Valor valor);
        /**
          * Cambia el criterio de ordenación y visualiza la lista
          */
        public void cambiarCriterioOrdenacion(String nombre, boolean orden);
        /**
          * Visualiza la lista de valores según el criterio de ordenación
          */
        public void visualizar();
}
```

```
public class Valor {
  private String nombre;
  private float precio;
  private float porcentaje;
  private float volumen;
  public String getNombre()
  public float getPrecio()
  public float getPorcentaje()
  public float getVolumen()
}
```

bolsa.txt

Valor1	16,4	1,1	200000
Valor2	5,6	3,5	150000
Valor3	4,2	2,4	100000
Valor4	6,4	2,2	175000
Valor5	10,2	1,8	300000

```
public class Token{
 /**
   * Abre el fichero de texto indicado por 'file'
   */
 public Token(String file);
 /**
   * Devuelve True si no se ha llegado al final de fichero y
   * False, en caso contrario
   */
 public boolean hasMoreTokens();
 /**
   * Devuelve los datos de un valor, correspondiendo a una
   * línea del fichero
   */
 public Valor getToken();
}
```

Se pide:

a) (0,75 punto) **Diseñar** e **implementar** la estructura (las variables miembro) más adecuada(s) para la clase *VerBolsa*, cuya misión es gestionar y visualizar la información de los valores, teniendo en cuenta las condiciones arriba descritas y las estructuras de datos vistas en clase. Implementa también el método constructor. **Nota**: En caso de necesitar algo más, implementa las clases y los métodos necesarios. Argumenta tus decisiones.

b) (1,5 punto) **Implementa** todos los métodos de la clase *VerBolsa*. Ten en cuenta las condiciones arriba descritas y tu respuesta en el apartado anterior.

c) (0,25 puntos) Calcula cual es la complejidad, en notación *O*, de los métodos implementados en b).

Nota: Se valorarán los aspectos de reutilización.

3. *Ejercicio (1,5 punto)*

Implementa el método *divideFrenteTrasera*, que dado una lista enlazada divide la lista en dos sublistas: una para la primera mitad (frente) y otra para la segunda mitad (trasera). Si el número de elementos es impar, entonces el elemento extra deberá ir en la primera mitad. Ejecutando el método sobre la lista {2, 3, 5, 7, 11} debería de devolver las dos siguientes listas {2, 3, 5} y {7,11} y la lista original queda vacía.

Pista. Probablemente el método más fácil es recorrer la lista para contar los elementos y posterior volver a recorrerlo hasta la mitad. Pero hay un truco que utiliza dos referencias para hacerlo en una pasada. Una de ellas, el "lento" avanza de uno en uno. En cambio el "rápido" avanza de dos en dos. Cuando el último llega al final de la lista, el "lento" estará por la mitad. En cualquiera de las dos opciones, hay que tener especial cuidado a la hora de dividir la lista en la posición adecuada. **Nota**: utiliza la segunda opción en tu implementación. Para devolver las dos listas utiliza la siguiente clase:

```
public class FrenteTrasera {
   public LinkedList frente;
   public LinkedList trasera;
}
```

Además, **describe** gráficamente los distintos casos que has tenido en cuenta y el momento principal de cómo se produce la división.

5. ***Ejercicio (3 puntos)***

Se desea realizar un sistema informático para escribir libros de cuentos dinámicos. Un *libro de cuentos dinámicos* tiene un titulo y está compuesto por varios capítulos, los cuales no se leen en el mismo orden de impresión. La característica principal del libro es que el lector es el creador de distintos cuentos (normales), los cuales tendrán varios capítulos del libro en común. La secuencia de los capítulos que forman un cuento (normal) la va estableciendo el lector al final de cada capitulo.

Un capítulo está compuesto por un titulo, seguido de un texto y al final, puede tener una pregunta o no. Todas las preguntas de cualquier capitulo siempre tendrán dos respuestas (sí o no). Además, cada respuesta indicará cual es el siguiente capítulo dónde continúa el cuento (los capítulos deben ser distintos), además de la página. De esta forma, dependiendo de las respuestas que vaya seleccionando el lector se creará un cuento u otro. Aquellos capítulos que no tengan una pregunta denotan el final de los cuentos.

La escritura de estos cuentos no es nada fácil. Como es de pensar, puede que algunas combinaciones de capítulos formen cuentos que no tengan ni pie ni cabeza y también, puede ocurrir que se formen ciclos, lo cual implica que se puede llegar a no terminar nunca un cuento. La prevención de crear este tipo de cuentos es una de las principales tareas del escritor.

La metodología que sigue el escritor para crear un libro de cuentos dinámicos es el siguiente:

1. Debido a la dificultad de escribir este tipo de cuento y para que no haya ciclos en el desarrollo de la secuencia del cuento, el escritor decide que todos los capítulos menos el de inicio solamente pueden tener un único capítulo como su predecesor en la secuencia de un posible cuento.

   Para ello, mientras va escribiendo los capítulos, el escritor prefija una posible secuenciación de capítulos mediante un esquema. Este esquema, llamado *esquema de cuentos dinámicos*, es una estructura arborescente (ver Figura 1) y sus objetivos principales son: *i)* organizar los capítulos que van a componer un libro de cuentos dinámicos en un orden determinado; *ii)* le sirva de ayuda al escritor para que escribir cuentos coherentes; y *iii)* cumpla las condiciones arriba descritas. Este esquema será su guía y lo mantendrá hasta que finalice el libro.

2. La edición de los cuentos dinámicos habrá finalizado, cuando no queden capítulos sin terminar. Un capitulo *está sin terminar*, si tiene pregunta y alguna de sus respuestas no tiene asignado el siguiente capitulo.

Para la representación del esquema de cuentos dinámicos se provee de las siguientes clases:

```
public class EsquemaCuentosDinamicos {
  private String titulo;
  private BinTree capitulos;
  private CapitulosBinTreeItr itrCapitulos;
  ...
}
```

```
public class Capitulo {
  private String titulo;
  private String texto;
  private String pregunta;
  public boolean tienePregunta();
  // y los correspondientes get y set
}
```

```
public class CapitulosBinTreeItr extends
BinTreeItr{
 ...
}
```

```
public class BTNode{
  protected Object content;
  protected BTNode left;
  protected BTNode right;
  // y los correspondientes get
}
```

```
public class BinTree{
  protected BTNode root;
  public BTNode getRoot();
}
```

```
public class BinTreeItr{
    protected BinTree bTree;
}
```

Nota: Cualquier otro método que actúe sobre estas clases tendrá que ser implementado.

ESTRUCTURA DE DATOS Y ALGORITMOS **6-Septiembre-2010**
Titulación: Ingeniero Técnico en Informática de Gestión **Curso: 2º**

Se pide:

Especificar, **diseñar** e **implementar** un método que dado un esquema de cuentos dinámicos completo (`EsquemaCuentosDinamicos`), representado mediante un árbol binario de capítulos, devuelva un vector (de cadenas de caracteres) con todos los distintos cuentos posibles del esquema completo. Cada cadena de caracteres describe un cuento, detallando el orden de los capítulos que forman el cuento y la respuesta (sí o no) seleccionada en la transición al capítulo siguiente.

```
public Vector imprimirCuentosIndividuales ()
```

La siguiente figura muestra un posible esquema de cuentos dinámicos completo:

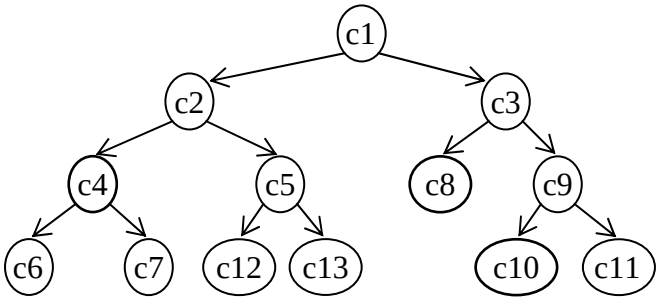

Figura 1. Árbol binario que representa un esquema de cuentos dinámicos. El nodo que no tiene ramas por debajo, significa que el capitulo no tiene pregunta, por lo tanto está terminado. Las ramas izquierdas representan la respuesta afirmativa a la pregunta del capitulo y la rama derecha, la negativa.

Para el ejemplo de la Tabla 1, la ejecución del método `imprimirCuentosIndividuales` devolvería un vector con el siguiente contenido:

cuento 1: C1--si-->C2--si-->C4--si-->C6
cuento 2: C1--si-->C2--si-->C4--no-->C7
cuento 3: C1--si-->C2--no-->C5--si-->C12
cuento 4: C1--si-->C2--no-->C5--no-->C13
cuento 5: C1--no-->C3--si-->C8
cuento 6: C1--no-->C3--no-->C9--si-->C10
cuento 7: C1--no-->C3--no-->C9--no-->C11