ESTRUCTURA DE DATOS Y ALGORITMOS **2-Septiembre-2008**
Titulación: Ingeniero Técnico en Informática de Gestión **Curso: 2º**

NOTA FINAL: Nota Practica (1 punto) + Nota Examen (9 punto)

- Es indispensable aprobar el examen (4,5 puntos) para sumar la puntuación de la práctica y para aprobar la asignatura es necesario una nota superior o igual a 5 puntos
- La práctica es opcional
- No está permitido el uso de apuntes, libros o móviles en el examen
- **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.**
- Duración: 3 horas
- Publicación de las notas: 8-9-2008
- Revisión del examen: 9-9-2008, 16.00-19:00

1. Ejercicio (1 puntos)

El array A contiene los elementos mostrados a continuación {3, 13, 8, 25, 45, 23, 98, 58}. Los dos primeros elementos se han ordenado utilizando el algoritmo `insertionSort`. ¿Cuál será el valor de los elementos del array después de tres pasadas más del algoritmo?

2. Ejercicio (1'5 puntos)

Implementar un método que dado un array de *`Persona`*s, ordene el array por la edad de la persona y en orden descendente, utilizando el algoritmo **genérico** `selectionSort`. La clase `Persona` contiene la información del nombre (cadena de caracteres) y la edad (entero). Implementa también el algoritmo `selectionSort`.

3. Ejercicio (1 punto)

Dado un array que contiene los elementos {8, 13, 17, 26, 44, 56, 88, 97} y utilizando el algoritmo de *búsqueda binaria*, **trazar** las etapas necesarias para encontrar el número 20.

4. Ejercicio. (2'5 puntos)

Añadir a la clase lista enlazada simple el método `removeCurrent()`. Este método elimina el nodo que indica la variable `current` (actual) y el nuevo valor de la variable `current` será el nodo siguiente del eliminado. El coste de ejecución de este método tiene que ser de orden constante *O(1)*.

Se pide:

a. **Definir** las clases necesarias para definir la estructura de listas enlazadas para almacenar cualquier tipo de contenido y las variables necesarias para implementar el método que se pide.
b. **Implementar** el método `removeCurrent()`.
c. Utilizando un ejemplo o varias, **realizar** para cada caso posible dos figuras que describan una lista enlazada y el estado de las variables utilizadas **antes y después de ejecutar** el método `removeCurrent()`.

5. Ejercicio (3 puntos)

El agente 0069 ha inventado un nuevo método de codificación de mensajes secretos. El mensaje original X se codifica en dos etapas:

a. X se transforma en X' reemplazando cada sucesión de caracteres consecutivos que no sean vocales por su imagen especular (es decir, al revés).
b. X' se transforma en la sucesión de caracteres X" obtenida al ir tomando sucesivamente: el primer carácter de X'; luego el último; luego el segundo; luego el penúltimo; etc.

Ejemplo: para X = *"Bond, James Bond"*, resultan:

X' = *"BoJ ,dnameB sodn"* y X" = *"BnodJo s, dBneam"*

Se pide:

a. **Diseña e implementa** los métodos de codificación y descodificación de mensajes, utilizando pilas y colas. Supón que el mensaje inicial viene dado como una cola de caracteres.
b. Analiza su complejidad y razona tu respuesta. ¿Hay alguna parte del proceso que se podría mejorar? En caso afirmativo, ¿Cómo lo mejorarías? Detalla cuales son las características (algoritmo más eficiente, nueva estructura de datos, operaciones más relevantes y su complejidad,…) de tu solución sin implementarla y calcula la nueva complejidad.

Nota: No se pueden utilizar las clases String ni StringBuffer o algo parecido.

Anexo I: Especificación de clases

```
class Character implements Comparable{
  Character(char c);//Es el método constructor de la clase (envoltorio) que representa un
                    // carácter como un objeto
  char getValue (); //pos: Devuelve el valor que esta envuelto, es decir, el carácter.
  int compareTo (Character c); // pos: Devuelve 0 si el objeto actual es igual a c,
                               // -1 si es menor que c y 1 si es mayor que c.
}
```