Examen. 29 de enero de 2013

No entregar esta hoja con el examen.
Todas las preguntas tienen la misma ponderación (25%).

1. Supongamos que tenemos definido el TAD genérico **Secuencia[T]** que almacena secuencias de elemento de tipo **T**, con las operaciones: **crear**, **añadir** (inserta un elemento al final de la secuencia), **último** (devuelve el último elemento de la secuencia), **principio** (devuelve la secuencia menos el último elemento); y disponemos del TAD **Natural** visto en clase.

   Incorporar al TAD **Secuencia[T]** las siguientes operaciones, todas ellas reciben como parámetros una secuencia y un elemento de tipo **T**:
   - **cuántas**: devuelve el número de apariciones del elemento en la secuencia dada;
   - **repetido**: sirve para saber si un elemento está repetido en la secuencia dada;
   - **eliminar-última**: elimina la última aparición del elemento dado en la secuencia dada;
   - **eliminar-rep**: elimina todas las apariciones, salvo la última, del elemento dado en la secuencia dada.

   Escribir las cláusulas NOMBRE y CONJUNTOS, así como la sintaxis y la semántica de las operaciones que se añaden. Se pueden añadir otras operaciones, que deberán ser especificadas también.

2. Como bien debes saber, un *anagrama* es una palabra que se obtiene permutando las letras de otra palabra (por ejemplo: amor, ramo, mora, roma…). En este problema definimos un tipo especial, que llamaremos *consonantegrama*, formado por todas las palabras que tienen las mismas consonantes, independientemente de las vocales (por ejemplo: esfera, ferias, foros, afosar, farsa…). Tenemos un diccionario y queremos organizarlo en *consonantegramas*. Para ello vamos a usar tablas de dispersión, con dos operaciones básicas: insertar una nueva palabra; y dada una palabra consultar de forma rápida sus *consonantegramas*. Describir todos los aspectos de la solución con dispersión: el tipo de dispersión, cómo se almacenan los datos, la función de dispersión, el tamaño de la tabla, las dos operaciones pedidas, etc.

3. En una aplicación tenemos implementados árboles B de orden **p** = *n*+1. El tipo ArbolB es básicamente un puntero a un nodo. Y los nodos del árbol son del siguiente tipo:

```
tipo NodoB = registro
                esHoja: booleano          // Indica si este nodo es una hoja
                numClaves: entero         // Indica el número de claves usadas
                claves: array [1..n] de T
                punteros: array [0..n] de Puntero[NodoB]
             finregistro
```

   Suponer que el tipo T se instancia a entero. Escribir una operación EscribirPares (raíz: Puntero[NodoB]), que dada la raíz del árbol, escribe por pantalla las claves con valor par, de manera ordenada (de menor a mayor).

4. Cierto programa en código máquina consta de *n* instrucciones. El programa empieza en la instrucción 1, y termina al llegar a la instrucción *n*. De cada instrucción sabemos cuántos ciclos de CPU que tarda: *T[i]* para *i*= 1…*n*. Las instrucciones pueden ser secuenciales, saltos, o saltos condicionales. Para las secuenciales y los saltos, *S1[i]* almacena el número de la siguiente instrucción, y *S2[i]* vale 0. Para los saltos condicionales, *S1[i]* y *S2[i]* indican los dos posibles lugares de salto (según se cumpla o no la condición).
   Escribir un algoritmo eficiente que calcule cuánto es el tiempo mínimo que, en teoría, puede tardar el programa en ejecutarse, y escriba la secuencia de instrucciones. Si existen varias soluciones posibles, debe indicar cuántas posibilidades existen.