Examen. 4 de septiembre de 2007

Resolver cada pregunta en una hoja distinta.
No hay que entregar esta hoja con el examen.

1. (2,5 puntos) Para la siguiente entrada: 2761, 8641, 2793, 8625, 1961, 2704, 8602, y dada una tabla de dispersión cerrada de tamaño $B = 10$ y la función de dispersión:

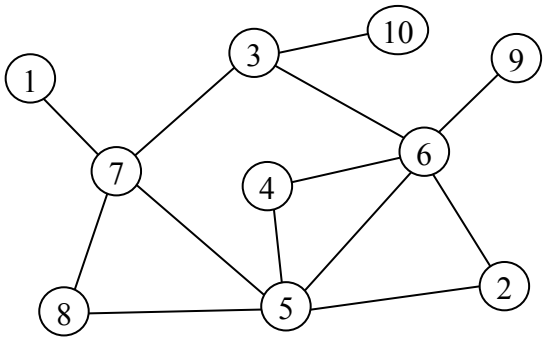
$$h(k) = \lfloor k/100 \rfloor \bmod B$$

   a) Mostrar la tabla resultante de insertar los anteriores elementos en la tabla usando redispersión lineal. Indicar cuándo se producen colisiones y cómo se resuelven.
   b) Lo mismo que el apartado anterior pero empleando redispersión cuadrática. ¿Es adecuada esta estrategia de redispersión para tablas de tamaño 10? Justificar la respuesta.
   c) Diseñar unas funciones de dispersión y redispersión distintas a las anteriores y que sean adecuadas para este ejemplo. Mostrar la nueva tabla resultante. ¿Crees que es necesario aplicar una reestructuración para este ejemplo concreto?

2. (2,75 puntos) Dado un árbol trie, programar una operación en pseudocódigo que encuentre las *n* primeras palabras en orden alfabético que aparecen en dicho árbol. Suponer que existe sobre el tipo trie una operación **Consulta** (n: trie, c: carácter): trie y un iterador del tipo **para cada** carácter c hijo del nodo n **hacer**. Se supone que este iterador recorre los hijos en orden alfabético. La cabecera de la función será: **operación** Primeras (n: entero; t: trie).

3. (2,75 puntos) ¡Viva la Feria de septiembre! Para celebrarlo vamos a encender una traca, que está compuesta de **n** petardos y trozos de mecha que los unen. Inicialmente se prende fuego a un único petardo, y la traca se extiende al resto de los petardos a través de los cabos de mecha. Todos los trozos de mecha son de la misma longitud.

   La matriz **C** de **n**x**n** de booleanos, indica en cada posición **C[i, j]** si los petardos **i** y **j** están conectados con una mecha. La relación es simétrica. Queremos que la traca dure lo menos posible en explotar. El objetivo es localizar a qué petardo hay que prenderle fuego inicialmente. Es decir, minimizar el tiempo en el que todos los **n** petardos han explotado. Escribir un algoritmo que encuentre cuál debe ser el petardo inicial, y que calcule el tiempo total de explosión. Aplicar el algoritmo al siguiente ejemplo.



4. (2 puntos) Suponer que tenemos las especificaciones formales de los TAD **Natural**, **Lista[T]** y **ArbolBinario[T]**, con las siguientes operaciones:
   - **N= Natural**: *cero*, *sucesor*, *esCero*, *esMenor*, *suma*, *resta*, *esIgual*.
   - **L= Lista[T]**: *vacía* (devuelve una lista sin elementos), *inserta* (dada una lista y un elemento, añade ese elemento en la primera posición de la lista), *concatenar* (une dos listas).
   - **A=ArbolBinario[T]**: *crear*, *construir* (crea un nuevo árbol dado un elemento y dos subárboles), *altura* (calcula la altura de un árbol, tomando 0 para el árbol vacío).

   Queremos añadir las siguientes operaciones al TAD **ArbolBinario[T]**: **quitarHojas** (dado un árbol, elimina todos los nodos hoja), **listarProfundidad** (dado un árbol, devuelve una lista que contiene los elementos del árbol listados en profundidad, es decir, en preorden), **listarAnchura** (dado un árbol, devuelve una lista con los elementos del árbol listados en anchura). Escribir la sintaxis y la semántica de estas operaciones. Se pueden incluir otras operaciones, pero será necesario especificarlas también.
   Sugerencia: para la última se aconseja añadir una operación *listarNivel(n, a)* (listar todos los nodos del árbol *a* en el nivel *n*) y *listarResto(n, a)* (listar todos los nodos de nivel *n*, luego los de nivel *n*+1, y así sucesivamente).

Nota: La pregunta 4 no deben hacerla los alumnos que tengan aprobada la Práctica de Especificaciones Algebraicas con Maude. Ojo, la asignatura "Laboratorio de Programación" no convalida esta práctica.

Examen. 4 de septiembre de 2007

Resolver cada pregunta en una hoja distinta.
No hay que entregar esta hoja con el examen.

1. (3 puntos).
   a) (1,5 puntos) Ordena las siguientes cotas de complejidad de menor a mayor, indicando también las que son iguales: $O(\log_5 n)$, $O(n^n)$, $O(3+5n+2n^2)$, $O(n!)$, $O(3\cdot 2^n)$, $O(20)$, $O(n\cdot \log \log n)$, $O(3^n)$, $O(n\cdot \log n)$, $O(8+ 2\cdot \log_2 n)$, $O(n(n+\log n))$, $O(1)$, $O(n2^n)$, $O(n+1)$, $O(2^n+3^n)$, $O(n^{n+1})$. Justifica brevemente los casos que no sean triviales. Indica por lo menos 8 ejemplos de algoritmos conocidos que tengan algunos de esos órdenes de complejidad. Por ejemplo, $O(n^3)$ → algoritmo clásico de multiplicación de matrices cuadradas de tamaño n.
   b) (1,5 puntos) Resuelve la siguiente ecuación de recurrencia, planteando el sistema de ecuaciones necesario para calcular las constantes que puedan surgir.

      $t(n)= 7t(n/5) – 10t(n/25) + 24n$ Si $n>30$ $t(n) = n-1$ Si $n<31$

      Expresar el orden de complejidad del tiempo de ejecución. En caso de haber realizado un cambio de variable, demostrar que se puede quitar la condición que aparezca en el orden.

2. (2 puntos) Tenemos un tablero de tamaño 2**n**+1 en el que hay **n** fichas blancas, **n** fichas negras y una posición vacía. En cada posición sólo puede haber una ficha, como máximo. Inicialmente, todas las fichas blancas están a la izquierda del tablero, y las negras a la derecha. El objetivo es, a través de una serie de movimientos permitidos, mover todas las fichas blancas a la derecha y las negras a la izquierda, como se muestra en la figura de abajo.

   Los movimientos permitidos son cuatro: (1) mover una ficha blanca a un hueco contiguo a la derecha; (2) mover una ficha negra a un hueco contiguo a la izquierda; (3) mover una ficha blanca a un hueco que esté dos posiciones a la derecha; y (4) mover una ficha negra a un hueco que esté dos posiciones a la izquierda.

   El problema consiste en encontrar una serie de movimientos que conduzcan de la situación inicial a la final. Programar en pseudocódigo un algoritmo para resolver el problema por avance rápido, e indicar las funciones que aparecen en el esquema. Si existen varias soluciones, cualquiera de ellas es válida.

3. (2,5 puntos) Resolver el problema anterior por backtracking. Habrá que indicarla forma del árbol que se recorre y las estructuras de datos que se utilizan para representar la solución. Utilizar el esquema de backtracking visto en clase, programando las funciones básicas del esquema (Solucion, MasHermanos, Criterio, Generar, ...). Sólo se busca una solución cualquiera.

4. (2,5 puntos) Resolver el problema del ejercicio 2 por ramificación y poda. En este caso, buscamos la solución que requiera el menor número de movimientos posible. Se deben utilizar los esquemas vistos en clase, que se pueden dar por supuestos. Definir la forma de representar la solución y todas las partes genéricas del esquema.

Nota: La pregunta 2 no deben hacerla los alumnos que tengan convalidada la Práctica 4 (entre ellos, los que tuvieran aprobada la asignatura "Laboratorio de programación" del plan antiguo).