Universidad de Alcalá

Prueba de evaluación continua
Estructuras de Datos
Todos los grados
Enero 2015

1.- (3 puntos) El Tipo Abstracto de Datos `MONTÍCULO[ELEMENTO]`.

- Explica qué es un montículo y para qué sirve. *(0'50)*
- Indica paso a paso las transformaciones del árbol al ir insertando los datos 7, 4, 8, 5, 2, 9, 6, 3 en un montículo de mínimos inicialmente vacío. *(0'75)*
- Usando la siguiente representación con memoria estática de un montículo:

  montículo = **reg**
  tamaño: 0..máximo
  datos: **vector**[1..máximo] **de** tipo_elemento
  freg

  implementa en pseudocódigo las operaciones que permitan crear un montículo y la inserción y borrado de datos, así como las operaciones auxiliares que sean necesarias. *(1'00)*
- Utilizando la representación y las operaciones del apartado anterior, implementa una función `ordenarHeapsort: vector → vector` para ordenar crecientemente un vector. *(0'75)*

2.- (3'75 puntos) Implementa las siguientes operaciones (pueden ser parciales, o necesitar otras funciones auxiliares) para el TAD `ÁRBOLES_BÚSQUEDA[ELEMENTO]` utilizando la representación con memoria dinámica siguiente:

nodo_a_bin = **reg**
valor: elemento
izq: a_bin
der: a_bin
freg

a_bin = **puntero a** nodo_a_bin

- `es_ABB?: a_bin → bool`, que comprueba si un árbol binario *a_bin* cualquiera cumple los requisitos para ser un árbol binario de búsqueda. *(1'00)*
- `poner: elemento a_bin → a_bin`, que inserta ordenadamente el *elemento* dentro del árbol de búsqueda *a_bin* pasado como parámetro. *(0'75)*
- `máximo: a_bin → elemento`, devuelve el mayor dato de un árbol de búsqueda. *(0'50)*
- `borrar_máximo: a_bin → a_bin`, para quitar el elemento más grande que se encuentre en un árbol de búsqueda. *(0'50)*
- `es_AVL?: a_bin → bool`, que comprueba si el árbol de búsqueda cumple los requisitos necesarios para ser un árbol AVL. *(1'00)*

3.- (3'25 puntos) Implementar en pseudocódigo las operaciones siguientes, usando las operaciones algebraicas de la especificación de árboles generales formados por letras `ÁRBOLES[LETRA]`:

- `listado_hojas: árbol → lista`, que genera una lista con todas las letras que se encuentran en las hojas del *árbol*, tomadas de izquierda a derecha. *(0'50)*
- `fila: árbol natural → lista`, que forma una lista con todas las letras del *árbol* que se encuentran en la profundidad indicada por el *natural*. *(0'75)*
- `más_repetida: árbol → letra`, que devuelve la letra que más veces aparece en el árbol. Para esta operación solo pueden utilizarse operaciones de árboles generales. *(1'00)*
- `está_palabra?: árbol lista → bool`, que comprueba si, comenzando en la raíz del *árbol*, existe un camino formado por las letras que aparecen en la *lista* indicada. *(1'00)*