Programación II. Parcial 2. 9 de mayo 2017 - Grupo de tarde

Apellidos (en mayúsculas):
_____Nombre:_____

Ejercicio 1 (2,5 puntos en total)

Dadas las estructuras de datos que hemos utilizado en clase para implementar un nodo y una lista enlazada circular:

En listaCircular.c:

```
struct _Nodo {
      Elemento *info;
      struct _Nodo *next;
};
typedef struct _Nodo Nodo;

struct _listaCircular {
      Nodo *last;
};
```

En listaCircular.h:

```
typedef struct _listaCircular listaCircular;
```

Implementa en C, **con control de errores**, la primitiva:

```
Status listaCircular_imprimir_vuelta(const listaCircular *pl, int pos, FILE *f);
```

que imprime en el fichero f los elementos de la lista a partir de la posición determinada por pos y dando la vuelta si es necesario. El valor pos=0 indica el primer elemento de la lista.

Ejemplo: si la lista contiene, empezando por el primer elemento, la secuencia A,B,C,D,E, al imprimir dando la vuelta dicha lista con pos=0 se imprimirá, en este orden, A,B,C,D,E.
Si pos = 2 se imprimirá C,D,E,A,B.
Si pos = 4 se imprimirá E,A,B,C,D.
Y si pos = 5 dará error sin imprimir nada.

Se podrá utilizar la siguiente primitiva del TAD elemento:

```
Status elemento_imprimir(Elemento *pe, FILE *f);
```

Programación II. Parcial 2. 9 de mayo 2017 - Grupo de tarde

Apellidos (en mayúsculas):
_____Nombre:_____

Ejercicio 2 (2,5 puntos en total)

(a) Proporcione el pseudocódigo de un algoritmo que determine si un árbol binario *tiene forma de V invertida*.

NOTA: Defineremos un arból binario tiene forma de V invertida como aquel árbol que o bien es vacío o bien todo nodo de su subárból izquierdo sólo tiene subárboles izquierdos y todo nodo de su subárbol derecho sólo tiene subárboles derechos.

(b) Prporcione el código C del algoritmo anterior asumiendo las siguientes macros y estructuras

```
#define ROOT(t) (t)->root
#define IZQ(pn) (pn)->izq
#define DER(pn) (pn)->der
typedef struct _Nodo {
   Elemento *info;
   struct _Nodo *izq;
   struct _Nodo *der;
} Nodo;

typedef struct _ARBOL ARBOL;
struct _ARBOL {
     Nodo *root;
}
```

Programación II. Parcial 2. 9 de mayo 2017 - Grupo de tarde

Apellidos (en mayúsculas):
_____Nombre:_____

Ejercicio 3 (2,5 puntos en total)

a) Crea, a partir de la siguiente expresión sufijo, su árbol de expresión (AdE) correspondiente, mostrando la evolución del algoritmo paso a paso: 3 4 2 / 1 5 * - + ;

b) ¿Qué información se puede obtener a partir de ese AdE, según el modo en que se recorra?

-

c) Define, en 2 o 3 líneas, qué es un árbol binario de búsqueda (ABdB). Dibuja uno que tenga más de 3 nodos.

d) ¿Qué información se obtiene recorriendo en orden simétrico (= medio) un árbol binario de búsqueda?

e) ¿Se puede crear un ABdB a partir de una secuencia de elementos ordenados?

☐ No → ¿Cómo tienen que estar los elementos necesariamente para poder crear el ABdB?

☐ Sí → Dibuja cómo sería la estructura del ABdB si esa secuencia tuviera 4 elementos ordenados de menor a mayor.

f) Muestra la evolución del algoritmo de creación de un ABdB a partir de la siguiente secuencia, dibujando, paso a paso, el estado del árbol que se va construyendo.

5 9 6 8 7

g) Observa la estructura del árbol obtenido en el ejercicio anterior (nodos, nº hijos de cada nodo, nº nodos en cada nivel, etc.). ¿Qué tipo de estructura tiene? _____

En este ABdB, ¿cuál es el coste de buscar un elemento, en cuanto al nº concreto de comparaciones a realizar…

… en el caso mejor?_____

… en el caso peor?_____

… en el caso medio?_____

h) Compara la utilización de los ABdBs en general (no necesariamente con la estructura del ejercicio anterior) para mantener ordenados y buscar elementos, frente a utilizar una lista enlazada simple en la que los elementos estén ordenados, en cuanto a:

- Eficiencia del procedimiento inicial hasta tener los elementos ordenados en la estructura.

- Eficiencia en la búsqueda

- Uso de memoria

Programación II. Parcial 2. 9 de mayo 2017 - Grupo de tarde

Apellidos (en mayúsculas):
_____Nombre:_____

Ejercicio 4 (2,5 puntos en total)

Star Wars: The Old Republic, es un videojuego de rol multijugador masivo en línea basado en el universo ficticio de Star Wars creado por George Lucas.

En él los usuarios (jugadores) desde sus ordenadores conectados a internet controlan personajes "avatares" dentro de un mundo virtual explorando el entorno, completando misiones, e interactuando y combatiendo con personajes no jugadores o con otros usuarios.

Para poder jugar en red los usuarios crean o se unen a una sesión de juego a través de un servidor de aplicaciones.

Un servidor mantiene un protocolo de comunicaciones que permite obtener y sincronizar los momentos (*timestamps*) en los que se realizan las acciones de los jugadores.

Suponer que en un intervalo de tiempo dado, un servidor recibe y procesa las siguientes peticiones de acciones de jugadores, ordenadas de izquierda a derecha según su llegada al servidor:

orden de llegada	1	2	3	4	5	6	7	8
jugador-acción	jedi1-A1	rebelde1-A2	sith1-A4	sith1-A2	stormtrooper1-A3	jedi2-A3	stormtrooper1-A1	jedi1-A1
timestamp	105	102	108	101	103	107	104	106

El servidor almacena las peticiones entrantes según el orden de llegada, y las procesa atendiendo a sus timestamps.

Así, en un momento dado, de las peticiones que tiene almacenadas, el servidor procesa la de menor timestamp.

Para ello, mantiene una cola de prioridad implementada con el TAD Heap en un array con estructura de datos.

A partir de las peticiones de la tabla de arriba:

- Crea un heap mediante el algoritmo "in-place". Muestra paso a paso el estado* del heap en cada paso del algoritmo. (1 punto)
- Tras haber almacenado las 8 peticiones, el servidor extrae del heap y procesa 2 peticiones. ¿Qué pares jugador-acción se ejecutan? Dibuja el estado* del heap tras cada una de esas extracciones. (1 punto)

(*) Para simplificar la visualización, representa cada nodo del heap con su timestamp.

En vez de un array, para almacenar el heap se plantea usar una estructura de datos ArbolBinario, compuesta de nodos NodoAB según se ha estudiado en la asignatura.

- ¿Qué diferencias en coste computacional de inserción/extracción y memoria empleadas tienen cada una de las dos estructuras de datos? (0,5 puntos)