Escuela de Ingeniería		
Examen de: Estructuras de datos y Algoritmos 1		**Código de materia: 1774**
Fecha: 19-10-2022	**Grupo: Todos**	**Hoja 1 de 2**
Duración: 2 horas		**Sin Material**

Ejercicio 1

Considere un árbol general de enteros representado mediante un árbol binario de enteros con la semántica: puntero al primer hijo (pH), puntero al siguiente hermano (sH).

```
typedef struct nodoAG * AG;
struct nodoAG { int dato; nodoAG *pH, *sH; }
```

Implemente la función **`int maximoHijos(nodoAG * raiz)`** que dado un árbol general retorna el valor máximo de número de hijos entre todos sus nodos. Si el árbol es vacío retorna cero. Asuma que existe una función `int cantHijos(nodoAG * raiz)`, que retorna la cantidad de hijos que dependen del parámetro raíz, que no puede ser vacío (es precondición).

Ejercicio 2

Considere la siguiente definición de listas de enteros:

```
typedef nodolista * Lista;
struct nodoLista { int dato; Lista sig; }
```

Implemente un procedimiento iterativo **`void quitarMayores(Lista &l, int x)`**, que dados una Lista y un entero "x", elimine de la lista todos los elementos mayores estrictos que el valor x. Si la lista es vacía o no contiene elementos mayores a x, el procedimiento no tendrá efecto.

Ejercicio 3

Una empresa quiere gestionar su stock de productos. Cada producto se identifica por un número entero no negativo y el precio del producto de tipo float. Pueden existir varias unidades de un mismo producto, por lo que también se registra la cantidad en existencia.

Para administrar las existencias se define el TAD Stock con la siguiente especificación:

```
struct representacionStock;
typedef representacionStock* Stock;

//POS: retorna un nuevo Stock vacío
Stock crearStock();

//POS: agrega el producto id al Stock si no existe.
//     Si existe modifica precio y cantidad
void agregar(Stock& s, unsigned int id, float precio, int cantidad);

//POS: elimina cantidad de unidades del producto id del Stock, si existe.
//     Si no existe, o la cantidad existente es menor, elimina el producto id.
void eliminar(Stock& s, unsigned int id, int cantidad);
```

```
//POS: muestra existencias en stock: id, precio y cantidad, ordenado por id
void listar(Stock s);

//POS: retorna la cantidad de unidades existentes del producto id
int cantidad(Stock s, unsigned int id);

//PRE: Cantidad(id)>0
//POS: retorna el precio asignado al producto id
float precio(Stock s, unsigned int id);

//POS: retorna la cantidad total de productos diferentes en Stock
int cantProductosDiferentes(Stock s);

//POS: destruye el Stock liberando memoria
void destruir(Stock& s);
```

Se pide:

a) Proponga una representación en la que la operación **`precio`**, se realice en O(log n) en caso promedio, siendo n la cantidad de productos diferentes en el stock y la operación **`cantProductosDiferentes`** se realice en tiempo constante en peor caso.

b) Desarrolle únicamente las operaciones **`agregar`**, **`listar`**, **`precio`** y **`cantProductosDiferentes`**, según la representación propuesta. Puede asumir implementadas las demás operaciones del TAD.

c) Sabiendo que el valor de los id de productos en stock está en el rango de [0..k], implemente una función **`float valorDelStock(Stock &s, int k)`**, que retorne el valor económico total del stock, considerando la suma de todos los productos existentes y multiplicando su precio por la cantidad existente de cada uno.