Cesta de la compra

Se desea diseñar e implementar una aplicación móvil que permita a un usuario crear y gestionar cestas de la compra cuando va a ir y está en el supermercado. Asumir que un usuario se registra en la aplicación y que ésta sólo le permite gestionar sus cestas de la compra y no las de otros usuarios.

Una cesta de la compra (identificada con un nombre) estará compuesta de una lista de artículos, cada uno de los cuales con una cantidad y unidad de medida concretas. Por ejemplo:

- 1 Kg. de plátanos
- 2 Kg. de manzanas
- 1 docena de huevos
- 1 botella de refresco de cola
- 1 caja de galletas

Los artículos están organizados atendiendo a una *jerarquía de categorías* de varios niveles:

```
FRUTAS
   Plátano
   Manzana
   ...
LÁCTEOS
   Huevos
   ...
BEBIDAS
   Agua
   REFRESCOS
      Refresco de cola
      ...
   ZUMOS
      Zumo de naranja
      ...
   ...
DULCES
   Galletas
   ...
...
```

Asumir que un artículo pertenece a una única categoría.

La aplicación ha de permitir al usuario explorar la jerarquía de categorías para encontrar los artículos.

Además, ha de permitirle mantener varias cestas de la compra, asociando cada una de ellas a un supermercado concreto.

Finalmente, para una cesta de la compra, en la aplicación el usuario puede marcar (o desmarcar) como "ya comprado" cada uno de sus artículos.

Con estas especificaciones, y asumiendo la existencia de tipos de datos genéricos (p.e. status, boolean) resuelve las siguientes cuestiones:

a) Listar y describir brevemente los TAD mínimos necesarios para almacenar la información de cestas de la compra y sus componentes (artículos con cantidades y unidades) y de artículos de la aplicación.

   TAD Categoria, para almacenar el nombre de una categoría y un puntero a su categoría padre en la jerarquía de categorías de la aplicación.

   TAD JerarquiaCategorias, para almacenar la jerarquía de categorías de la aplicación.

   TAD Articulo, para almacenar la información (nombre y categoría) de un artículo.

TAD CestaCompra, para almacenar los 'componentes' de una cesta de la compra.

TAD ComponenteCestaCompra, para almacenar un 'componente' de una cesta de la compra, compuesto de un artículo y del número de elementos de ese artículo en la cesta, p.e. 2 Kg. de manzanas.

b) A partir de los TAD del ejercicio anterior, escribir las estructuras de datos C necesarias para su implementación. En las estructuras se pueden usar arrays estáticos en vez de punteros para memoria dinámica. Para cada TAD, dar los nombres de los ficheros .c y .h correspondientes indicando qué información sobre las estructuras contendrían.

```
// En categoria.h
typedef struct _Categoria Categoria;

// En categoria.c
struct _Categoria {
          char nombre[MAX_NOMBRE];
          Categoria *categoriaPadre;
};

// En jerarquia.h
typedef struct _JerarquiaCategorias JerarquiaCategorias;

// En jerarquia.c
struct _JerarquiaCategorias {
          Categoria *categorias[MAX_CATEGORIAS];
          int numCategorias;
};

// En articulo.h
typedef struct _Articulo Articulo;

// En articulo.c
struct _Articulo {
          char nombre[MAX_NOMBRE];
          Categoria *categoria;
};

// En componenteCesta.h
typedef struct _ComponenteCestaCompra ComponenteCestaCompra;

// En componenteCesta.c
struct _ComponenteCestaCompra {
          Articulo *articulo;
          int cantidad;
          char unidad[MAX_UNIDAD];
}

// En cesta.h
typedef struct _CestaCompra CestaCompra;

// En cesta.c
struct _CestaCompra {
          char nombre[MAX_NOMBRE];
          char supermercado[MAX_NOMBRE];
          ComponenteCestaCompra *articulos[MAX_ARTICULOS];
          int numArticulos;
}
```

c) Escribir el prototipo de una función C que imprima en un flujo de salida dado los datos de un artículo.

```
status articulo_imprimir(FILE *salida, const Articulo *articulo);
```

d) Escribe el código C de una función que imprima una cesta de la compra dada con un formato similar al del ejemplo del enunciado y gestionando errores de ejecución.

En cesta.c:

```
status cesta_imprimir(FILE *fsalida, const CestaCompra *pcesta) {
          int ret, i;
          status st;

          if (!fsalida || !pcesta)
             return ERROR;

          ret = fprintf(fsalida, "Cesta %s:\n", pcesta->nombre);
          if (ret != 1)
             return ERROR;

          ret   =   fprintf(fsalida,   "Supermercado:   %s\n",   pcesta-
>supermercado);
          if (ret != 1)
             return ERROR;

          fprintf(fsalida, "%d artículos:\n", pcesta->numArticulos);
          for (i=0; i<pcesta->numArticulos; i++) {
               st    =    componenteCesta_imprimir(fsalida,    pcesta-
>articulos[i]);
               if (st == ERROR)
                  return ERROR;
          }

          return OK;
}
```

En componenteCesta.c:

```
status componenteCesta_imprimir(FILE *fsalida, const componenteCesta
*pccesta) {
          int ret;
          status st;

          if (!fsalida || !pccesta)
             return ERROR;

          ret = fprintf(fsalida, "\t- %d %s de ", pccesta->cantidad,
                    pccesta->unidad);
          if (ret != 2)
             return ERROR;
```

```
            st = articulo_imprimir(fsalida, pccesta->articulo);
            if (st == ERROR)
               return ERROR;

            return OK;
}
```