Programación para Sistemas. EXAMEN PRÁCTICO PROGRAMACIÓN C
Ejercicio Evaluación Progresiva
24 de Enero de 2023. Duración: 90 minutos

El ejercicio consiste en escribir un programa en lenguaje C que procese una matriz con *NC* datos de *NM* teléfonos móviles (en adelante, terminales). Observar que la situación es equivalente a que los datos se lean de un fichero de texto donde cada línea es una fila y cada campo de datos una columna. El orden de los campos es el siguiente:

Cada línea del array contiene los siguientes campos:

- IMEI: código de identificación del terminal
- Model: descripción comercial del modelo
- RAM: memoria (Gb) del terminal
- Bat: autonomía (mAh) del terminal
- PVP: precio (€) de venta al público
- Spec: un valor entero con el n.º de caracteres contenidos en las especificaciones detalladas del terminal móvil

Para la realización del ejercicio es fundamental **leer con atención el código de apoyo**. En ningún caso, se modificará siendo preciso únicamente escribir el código de las funciones solicitadas en los archivos ***movil.c*** y ***vect_movil.c***

El programa deberá procesar una matriz de punteros al tipo *char* convirtiendo cada fila en un *struct* que se define en la cabecera ***movil.h*** (tipo ***movil_t***). Cada elemento en la matriz se deberá convertir al tipo de datos del campo correspondiente. *NM* y *NC* se definen como constantes simbólicas en la cabecera ***vect_movil.h*** porque, si bien el número de campos (la constante *NC*) es fijo, se debe contemplar la posibilidad de que cambie el valor de *NM* (el número de terminales)

El otro requisito fundamental del programa es que existe un límite ***n*** para el número de datos del tipo ***movil_t*** que simultáneamente pueden permanecer en memoria. Este valor se recibe como argumento en la línea de orden y determina el tamaño del vector de móviles que se pide en ***vect_movil.c***

El programa deberá mostrar en pantalla los ***n*** últimos terminales en la matriz de entrada donde ***n*** coincide con el límite anterior. Los terminales se visualizarán en orden inverso, esto es, primero el último. Se incluye como parte del material de apoyo un fichero denominado *test_movil.txt* que muestra el resultado para n=5 terminales

Se suministran dos ejemplos en un programa de test denominado ***test_movil.c*** que muestran la utilización de las funciones pedidas en ***movil.c*** y en ***vect_movil.c***:

1. En el primero la función ***mov_test()*** muestra el uso de ***new_movil()***, ***del_movil()*** y de ***toString()*** con un array ***asfmt[6]*** con las cadenas de formateo correspondientes a los 6 campos del tipo ***movil_t***. Se muestra también la longitud del *string* resultante
2. El segundo muestra la invocación de ***vect_movil()*** con los siguientes argumentos:
    - el número máximo de terminales que pueden permanecer simultáneamente en memoria. Este valor se toma de los argumentos en la línea de orden
    - un array de ejemplo de cadenas de formateo denominado ***sfmt***
    - una matriz de ejemplo denominada ***vmov*** con datos de terminales

Se pide:

movil.c

```
movil_t *new_movil(const char *imei, const char *modelo, const char *ram, int bat, double pvp, int sspec)
```
[2p]

- Crea un dato del tipo ***movil_t*** y almacena los datos en los argumentos de la función en los campos correspondientes de la estructura
- Esta función se usará para crear un terminal antes de almacenarlo en el vector de dimensión ***n***

```
void del_movil(movil_t *pm)
```
[1p]

- Libera la memoria de un dato ***movil_t***
- Esta función se usará para eliminar un terminal del vector de dimensión **n**

```
char *toString(movil_t *pmov, const char *sfmt[], char *mstr)
```
[3p]

- Devuelve una cadena de texto que contiene los datos de los campos del terminal en el primer argumento formateados según se indica en el segundo argumento ***sfmt***. Si algún elemento de ***sfmt*** es NULL, el campo correspondiente se ignora
- En la implementación se utilizará ***sprintf()***. Esta función es análoga a ***printf()*** con la diferencia de que en lugar de escribir en la salida estándar escribe sobre un buffer donde se va a almacenar la cadena de texto formateada. El argumento ***mstr*** es, precisamente, la dirección de dicho buffer
- Por consiguiente, se debe garantizar que antes de llamar a ***toString()*** existan ***MAXB*** caracteres de memoria asignada para almacenar la cadena. Si la longitud de la cadena resultante fuera menor que ***MAXB*** entonces se debe redimensionar con ***realloc()*** para ocupar únicamente la memoria estrictamente necesaria

vect_movil.c

```
int vect_movil(int n, const char *sfmt[],
               const char *vmov[NM][NC])
```
[4p]

- Procesa la matriz con los datos de los móviles que recibe en el argumento ***vmov***
- La función debe cumplir con el requisito de asignar memoria para un máximo número de terminales indicado por el argumento ***n***
- Los terminales se crean invocando a ***new_movil()***
- Para mostrar en la salida estándar los n últimos móviles invoca ***toString()***. Si no se ha implementado o no funciona ***toString()*** se podrá usar ***printf()***
- Antes de terminar libera toda la memoria que hubiera reservado

IMPORTANTE: No se modificará ningún archivo de cabecera ni tampoco el código suministrado. Únicamente se entregarán los archivos ***movil.c*** y ***vect_movil.c***. Se podrá utilizar cualquiera de las llamadas al sistema en los includes que aparecen en los esqueletos

printf(): https://man7.org/linux/man-pages/man3/printf.3p.html

sprintf**():** https://man7.org/linux/man-pages/man3/sprintf.3p.html

realloc**():** https://man7.org/linux/man-pages/man3/realloc.3p.html