Programación de Sistemas y Concurrencia

Examen 2ª convocatoria ordinaria
Curso 2020-2021

APELLIDOS_____ NOMBRE_____

DNI _____ ORDENADOR _____ GRUPO/TITULACIÓN_____

Bloque 1 – Programación de Sistemas

Descripción del sistema

Una compañía ferroviaria quiere gestionar el uso de los asientos de sus trenes. Para ello, en cada momento, almacena los asientos ocupados y el destino de su pasajero. El objetivo de este ejercicio es implementar un módulo C con las estructuras y funciones necesarias para gestionar los asientos de un tren del modo indicado. Se utiliza la siguiente definición de tipos, incluida en el fichero `GestionTren.h`:

```
#define MAX_DESTINO 20
typedef struct  Pasajero *Tren;
struct Pasajero{
        int asiento;
        char destino[MAX_DESTINO];
        struct Pasajero *sig;
};
```

La información de los pasajeros que viajan en un tren se almacena en una lista enlazada (de nodos de tipo **struct Pasajero**) ordenada de forma creciente por el número de asiento que ocupa cada pasajero. La siguiente figura muestra el estado del tren en un momento en el que hay 4 pasajeros.

Además del fichero de cabecera (`GestionTren.h`), se proporciona un fichero para que probéis la implementación (`Driver.c`), y un ejemplo de la salida por consola al ejecutar el fichero de pruebas (`driver_output.txt`). Se pide implementar las siguientes funciones en el fichero `GestionTren.c` (**es el fichero que debéis entregar**):

```
/* 0.5 ptos: al salir de la cochera el tren está vacío (equivale a inicializar el tren) */
Tren salirCochera();

/* 0.5 ptos: Mostrar por pantalla los pasajeros que están en el tren.
 *           Hay que mostrar el asiento y destino de cada uno */
void mostrarPasajeros(Tren t);

/* 1 pto: al entrar a la cochera tienen que bajarse todos los pasajeros que haya en ese momento en el tren,
 *        independientemente de cuál sea su destino.
 *        Devuelve el número de pasajeros que se han bajado en la cochera */
int entrarCochera(Tren *t);

/* 2 ptos: un pasajero intenta subir al tren.
 *         Si el asiento está disponible se almacena en el tren el asiento y destino del pasajero y se devuelve 1.
 *         Si el asiento está ocupado no se modifica la lista y se devuelve 0. */
```

```
int subirPasajero(Tren* t, int asiento, char *destino);

/* 2 ptos: cuando el tren llega a una parada, se bajan del tren (se extraen de la lista)
 *          todos los pasajeros con ese destino.
 *          La función devuelve el número de pasajeros que se han bajado */
int parada(Tren *t, char *parada);

/* 2 ptos: Almacena el estado del tren en un fichero binario cuyo nombre se recibe como parámetro.
 *          Por cada pasajero del tren hay que almacenar: el número del asiento, la longitud del nombre del
 *          destino y el nombre del destino (solo los caracteres correspondientes a su longitud). */
void guardarBinario(Tren t, char *fichero);

/* 2 ptos: Crear un tren con los datos almacenado en el fichero binario cuyo nombre se recibe como parámetro.
 *          El fichero tendrá el formato generado por la función guardarBinario, es decir que por cada pasajero hay
 *          que leer su asiento, la longitud del nombre del destino, y el nombre del destino (solo los caracteres
 *          correspondientes a su longitud). */
Tren cargarBinario(char *fichero);
```

ANEXO

Los prototipos de las funciones para manipular strings (incluidas en <string.h>) son:

char* strcpy(char *s1, char *s2): Copia los caracteres de la cadena s2 (hasta el carácter '\0', incluido) en la cadena s1. El valor devuelto es la cadena s1.

int strcmp(char *s1, char *s2): Devuelve 0 si las dos cadenas son iguales, <0 si s1 es menor que s2, y >0 si s1 es mayor que s2.

size_t strlen(const char *s): Calcula el número de caracteres de la cadena apuntada por s. *Esta función no cuenta el carácter '\0' que finaliza la cadena*.

Los prototipos de las funciones para **manipulación de ficheros binarios** (incluidos en <stdio.h>) son los siguientes (se dan por conocidos los prototipos de las funciones de <stdlib.h> que necesites, como free o malloc):

FILE *fopen(const char *path, const char *mode): Abre el fichero especificado en el modo indicado ("rb"/"wb" para lectura/escritura binaria y "rt"/"wt" para lectura/escritura de texto). Devuelve un puntero al manejador del fichero en caso de éxito y NULL en caso de error.

int fclose(FILE *fp): Guarda el contenido del buffer y cierra el fichero especificado. Devuelve 0 en caso de éxito y -1 en caso de error.

unsigned fread(void *ptr, unsigned size, unsigned nmemb, FILE *stream): Lee nmemb elementos de datos, cada uno de tamaño size bytes, desde el fichero stream, y los almacena en la dirección apuntada por ptr. Devuelve el número de elementos leídos.

unsigned fwrite(const void *ptr, unsigned size, unsigned nmemb, FILE *stream): Escribe nmemb elementos de datos, cada uno de tamaño size, al fichero stream, obteniéndolos desde la dirección apuntada por ptr. Devuelve el número de elementos escritos.