Facultad de Informática – Universidad Complutense de Madrid
Fundamentos de la programación I – Grupos A, B, I y DG-ADE+II

Examen convocatoria extraordinaria 2019-2020

Tiempo disponible: 2 horas y 30 minutos

Se pide desarrollar un programa que permita actualizar la clasificación de una liga no profesional de futbol con *n* equipos, siendo *n* un entero par (siempre menor o igual a un cierto valor MAX_EQUIPOS=20). Para cada equipo de la liga dispondremos de un acrónimo único (una cadena de caracteres sin espacios en blanco) y del número de puntos obtenidos hasta el momento. Por ejemplo, si en la liga hay 10 equipos podríamos tener una información como la siguiente:

Celta	GET1	ATM	DEP	BCN	GET2	ZARA	Sevilla	VAL	RM
42	12	50	31	53	14	16	12	14	49

La clasificación se encuentra almacenada en el fichero `liga.txt`. En dicho fichero, la información de cada participante se encuentra en una línea: primero el acrónimo y después el número de puntos, separados por un espacio en blanco. El fichero termina con el centinela FIN, y el número de equipos para los que se ha guardado información en él será siempre par y menor o igual que MAX_EQUIPOS. A la derecha se puede ver el contenido de este fichero para los datos anteriores.

```
Celta 42
GET1 12
ATM 50
DEP 31
BCN 53
GET2 14
ZARA 16
Sevilla 12
VAL 14
RM 49
FIN
```

La información de la clasificación se actualiza mediante los datos del fichero `jornada.txt`: los resultados de los encuentros celebrados. Este fichero contiene exactamente un número de líneas igual a la mitad de equipos que se hayan cargado desde el archivo `liga.txt`. Cada línea contiene 4 datos *equipo1 entero1 equipo2 entero2* separados entre sí por un espacio en blanco, denotando que los equipos *equipo1* y *equipo2* han disputado un partido y el *equipo1* ha marcado *entero1* goles mientras que el *equipo2* ha marcado *entero2* goles. A la izquierda se puede ver un ejemplo de fichero `jornada.txt`.

```
ATM 5 Celta 0
DEP 3 BCN 1
GET2 4 GET1 4
RM 0 ZARA 9
VAL 1 Sevilla 1
```

Al procesar una jornada, si un equipo ha marcado más goles que el contrario recibirá 3 puntos, recibiendo el contrario 0 puntos. Si los dos equipos han marcado el mismo número de goles entonces cada equipo recibe 1 punto. Por ejemplo, si procesamos la jornada anterior para la clasificación indicada, deberíamos obtener una nueva clasificación:

Celta	GET1	ATM	DEP	BCN	GET2	ZARA	Sevilla	VAL	RM
42	13	53	34	53	15	19	13	15	49

En primer lugar, empieza declarando los **tipos y constantes** adecuados para representar en memoria un equipo –tipo `tEquipo`– (acrónimo y puntos), la clasificación de los equipos de la liga –tipo `tLiga`– (lista de hasta MAX_EQUIPOS equipos), y los resultados de la `jornada` –tipo `tJornada`– **(1,5 puntos)**.

El programa principal comenzará cargando los datos de la clasificación y de la jornada. Si la carga de la clasificación y/o de la jornada falla, se mostrará un mensaje de carga fallida y finalizará la ejecución del programa. En caso contrario se mostrará por pantalla la clasificación; se actualizará la clasificación con los datos de la jornada; se volverá a mostrar la clasificación; y finalmente se mostrarán por pantalla al primer y al último clasificado.

Debes implementar los subprogramas siguientes:

- **(1,5 puntos)** `cargarLiga` que carga la clasificación actual a partir del fichero `liga.txt`. Devuelve la clasificación de tipo `tLiga`, así como un valor booleano que indica si la tarea se pudo llevar a cabo o no (el fichero podría no existir).
- **(1 punto)** `cargarJornada` que carga la jornada a partir del fichero `jornada.txt`. Recibe el número de equipos de la clasificación y devuelve la jornada de tipo `tJornada`, así como un valor booleano que indica si la tarea se pudo llevar a cabo o no (el fichero podría no existir).
- **(1 punto)** `mostrarLiga` que recibe una clasificación de tipo `tLiga` y la muestra por pantalla (la información de cada equipo en una línea).
- **(3 puntos)** `actualizarLiga` que recibe una clasificación de tipo `tLiga` y una jornada de tipo `tJornada`, y modifica los datos de la clasificación de acuerdo con los resultados obtenidos en la jornada según se ha explicado previamente. Se puede asumir que ningún equipo aparece dos o más veces en la jornada y que los acrónimos de los equipos de la jornada son válidos (coinciden con los de algún

equipo de la clasificación). Además del subprograma `actualizarLiga` debes implementar el subprograma `buscar`, que dada la clasificación y el acrónimo de un equipo devuelva la posición en la que se encuentra el equipo en la clasificación.

- **(2 puntos)** `primeroYultimo` que recibe una clasificación de tipo `tLiga` y devuelve los acrónimos del primer y del último clasificado, es decir, del que lleva más puntos y del que lleva menos. Si hay varios equipos que puedan ser el primero (por tener el mayor número de puntos de todos los equipos de la clasificación y estar empatados a puntos) entonces se escoge el que esté más cerca del principio de la lista. Si hay varios equipos que puedan ser el último (por tener el menor número de puntos de todos los equipos de la clasificación y estar empatados a puntos) entonces se escoge el que esté más cerca del final de la lista. Por ejemplo, para la clasificación actualizada del ejemplo, el primer clasificado sería `ATM` (`ATM` y `BCN` tienen el mayor número de puntos, pero `ATM` está más cerca del principio de la lista que `BCN`) y el último clasificado sería `Sevilla` (`GET1` y `Sevilla` tienen el menor número de puntos, pero `Sevilla` está más cerca del final de la lista que `GET1`).

Recuerda que en el código que desarrolles se valorarán la corrección del diseño, la eficiencia del código, la ausencia de código repetido, la correcta utilización de los esquemas vistos en clase, etc. Recuerda que no se permite el uso de variables globales, ni de instrucciones de salto salvo return en las funciones.