UNIVERSIDAD CARLOS III DE MADRID
PROGRAMACIÓN. GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES
CONVOCATORIA EXTRA ORDINARIA JUNIO 2014. PROBLEMAS

Apellidos:_____
Nombre:_____

Segunda parte. Problemas. Duración 2h 30 min.
Problema 1. (3,5 puntos del total del examen)

Se desea diseñar un programa en C para realizar una simulación de la Copa Mundial de Fútbol que se está celebrando actualmente en Brasil. Vamos a suponer que en el torneo participan un total de 16 selecciones. Cada una de ellas se modela mediante una estructura de datos que contiene la siguiente información: identificador de selección (número entero entre 1 y 16), nombre del país, cabeza de serie (número entero que indica si esta propiedad es verdadero o falso), coeficiente FIFA (número real que indica lo buena que es una selección), y un vector con los datos de los 20 jugadores convocados.
Cada jugador se modela mediante una estructura de datos que contiene la siguiente información: identificador del jugador (número entero entre 1 y 20 que coincide con su dorsal), nombre completo y número de goles marcados durante el campeonato.
Se pide:

a) Definir las estructuras necesarias para representar las composiciones de datos descritas. [1 punto]

b) Definir una función que solicite por pantalla los datos de una selección, incluido el coeficiente FIFA. El número de goles marcados por cada jugador debe inicializarse a 0 y al identificador de selección debe asignársele el valor dado por el parámetro formal *numsel*. Además, las selecciones con identificador entre 0 y 3 serán cabeza de serie. La función debe tener la siguiente cabecera [1.5 puntos / 10]:

```
void anadir_seleccion(struct seleccion selecciones [MAXSELE], int numsel);
```

c) Para disputar el torneo, se ha decidido dividir inicialmente las selecciones en NUMGRUPOS grupos de NUMSELECCIONES selecciones, que vamos a representar en nuestro programa mediante una matriz con el identificador de las selecciones y de tamaño NUMGRUPOS x NUMSELECCIONES. Cada celda contendrá el identificador de una selección, y cada columna corresponderá con un grupo. Se pide generar el código de una función que genere automáticamente esta matriz teniendo en cuenta los siguientes requisitos [1.5 puntos/ 10]:
   1) Cada grupo sólo debe contener una selección cabeza de serie (existen únicamente 4 en el campeonato).
   2) Cada selección sólo puede estar en un grupo.

   La función debe tener la siguiente cabecera:

```
void generar_grupos(struct seleccion selecciones [MAXSELE], int grupos [NUMGRUPOS][NUMSELECCIONES]);
```

d) Diseñar una función que determine el vencedor de cada grupo en la fase inicial. Se considera que el vencedor de un grupo es directamente la selección del grupo que tenga un mayor coeficiente FIFA en el campo correspondiente. [2 puntos/ 10].

```
void ganadores_fase_inicial (struct seleccion selecciones [MAXSELE], int grupos [NUMGRUPOS][NUMSELECCIONES],
                             int ganadores_fase1 [NUMGRUPOS]);
```

e) Suponiendo que al final de la primera fase del campeonato se hubiese actualizado el número de goles marcados por cada jugador, se pide diseñar una función que ordene los jugadores de una selección en función del número de goles marcados. [2 puntos].

```
void ordenar_jugadores(struct seleccion sel, struct jugador jugadores_ordenados[MAXJUG]);
```

f) Por último, diseñe la función principal del programa de acuerdo con el siguiente conjunto de pasos [2 puntos/ 10]:
   1. Requerir al usuario que complete los datos de las selecciones que participan en el campeonato.
   2. Realizar una simulación de la fase inicial del campeonato (creación de la matriz de grupos y determinación del ganador de cada uno de ellos).
   3. Mostrar el nombre de la selección ganadora de cada grupo en la fase inicial.
   4. Mostrar el nombre del máximo goleador de las selecciones ganadoras de cada grupo en la fase inicial.

Problema 2. (3,5 puntos del total del examen)

Se quiere programar una aplicación *e-learning* para ayudar en el aprendizaje de un conjunto de temas. La aplicación realiza al usuario una serie de preguntas (que nunca se repiten), organizadas en tandas de 2 preguntas por tema. Cada pregunta sólo puede tener por respuesta *verdadero* o *falso*. El programa comienza con una tanda de 2 preguntas del tema 1. Si el usuario responde correctamente las dos preguntas se pasa a una nueva tanda del tema siguiente, pero si falla alguna, se le vuelve a presentar una tanda del primer tema, hasta que responda bien una tanda completa. Lo mismo sucede en el tema 2 y siguientes, con la diferencia de que en estos temas, si se fallan las dos respuestas en una tanda, se sitúa al usuario en el tema anterior hasta que lo supere contestando bien las dos preguntas de una tanda. Si contesta una bien y otra mal, la siguiente tanda será del mismo tema.
El programa termina cuando se hayan respondido bien las dos preguntas de una tanda del último tema. Al terminar, el programa comunicará al usuario la nota obtenida en cada tema y le recomendará repasar los temas en los que la nota obtenida sea menor que una nota de corte. Tras cada tanda de preguntas se da al usuario la opción de terminar el programa, en cuyo caso, por ser una salida forzada, no se le comunica la nota ni la recomendación.

La nota de un tema se calcula como 10 x aciertos / preguntas hechas

El programa utiliza una llamada a la función *cargarPreguntas,* que se encarga de rellenar las fichas de preguntas, cuyo prototipo es el siguiente:

void cargarPreguntas (struct fichaPregunta cuestiones[]);

Esta función se considera que ya está disponible, por lo que bastará con insertar en las directivas del preprocesador la línea <include preguntas.h>

Se pide:

1. Definir las estructuras de datos: **(1 punto /10)**
    a. Ficha de pregunta, con tres campos: número de tema, texto de la pregunta y respuesta (1: verdadero o 0: falso).
    b. Vector de fichas de preguntas, que almacene todas las preguntas posibles
    c. Ficha de resultados del usuario en un tema, con tres campos: número de cuestiones preguntadas, número de contestadas bien, y nota.
    d. Vector de fichas de resultados, que almacene los resultados del usuario en cada tema (el programa está diseñado para que haya un único usuario)
2. Escribir la función *elegirPregunta,* que tiene como parámetros formales el número de tema, un vector de fichas de preguntas y un vector de preguntas usadas. La función buscará la primera pregunta no usada del tema y devolverá su posición en el vector que contiene las fichas de preguntas. **(2 puntos /10)**
3. Escribir la función *preguntarTema,* que tiene como parámetros formales el número de tema, un vector de fichas de preguntas, un vector de preguntas usadas y un vector de fichas de resultados. La función *preguntarTema* debe llamar a la función *elegirPregunta* para elegir qué preguntas hacer, formularlas al usuario y actualizar el número de respuestas correctas en un determinado tema. **(3 puntos /10)**
4. Escribir la función *main* que contendrá el bucle principal para subir o bajar de tema, y llamará a *cargaPreguntas, preguntarTema, y calificar* y *recomendar.* **(3 puntos /10)**
5. Escribir las funciones *calificar y recomendar.* Calificar calcula la nota de cada tema y la muestra por pantalla, y recomendar muestra la lista de temas que se deben repasar. **(1 puntos /10).**

Nota: el número máximo de preguntas y de temas y la nota de corte se definirán como constantes. Se les puede dar el valor que se desee.