Referencias:

a. Los puntos 1 a 4, se refieren a la carga de datos, en el ejemplo están resueltos de forma estática en el caso de los struct, y para el caso de los números, con la función rand (hay un ejemplo de esta función en foro). Esto facilita la prueba sobre el resto de las funciones.
b. Los valores de N y M (o fila, columna) quedan a criterio propio, para este ejemplo se utiliza: N = 4, M = 6
c. No hay que utilizar variables globales.
d. El ejercicio no tiene struct anidados.
e. Los nombres de los campos de los esquemas de los struct son aclaratorios, se recomienda simplificarlos.
f. El ejercicio recorre los últimos temas, arrays de una y dos dimensiones, struct y ordenamientos, sin búsquedas.
g. Una vez que los datos a procesar están cargados, hay que resolver los 5 puntos restantes.
h. A partir del punto 5, explico cada punto para disminuir las dudas, no obstante pueden consultarnos.

Datos:

1. Una organización de torneos de natación, tiene inscriptos N competidores, cada uno representa provincias del país. Para ello, se define un array de N struct con los siguientes datos:

   Id_competidor – tipo int
   Provincia - array de tipo char[30]
   Tiempos – array de tipo float[10]
   Promedio_de_tiempos_del_mes - tipo float ***(inicialmente campo en 0)***
   Maximo_Promedio - de tipo float ***(inicialmente campo en 0)***

2. Además se define un array con N datos de instructores responsables:

   Id_competidor - tipo int
   Nombre_del competidor– array de tipo char[30]
   Nombre_del instructor – array de tipo char[30]

3. Y se define un tercer array de enteros de dimensión N que se inicializará con – 1.
4. Por otra parte, se almacenan en una array bidimensional de N filas por M columnas, los promedios mensuales de los últimos 6 meses por competidor:

	m_1	**m_2**	**m_3**	**m_4**	**m_5**	**m_6**
c_1	167	150	180	212	158	233
c_2	405	245	334	253	201	154
c_3	151	209	143	453	154	321
c_4	182	193	452	123	183	149

Requerimientos:

5. Función que reciba el array de competidores y cargue en el campo '*promedio_de_tiempos_del_mes'*, el promedio del campo del struct de 10 tiempos. ***Explicación:*** *el array de competidores original tiene 2 campos que están vacíos y campo array de 10 tiempos del mismo struct, por lo tanto esta función recorre cada posición con su array de 10 tiempos para obtener el promedio. Una vez obtenido el promedio se guarda en el campo 'promedio_de_tiempos_del_mes'. Entonces completamos ese campo vacío.*

6. Función que reciba el array de competidores, la matriz de tiempos mensuales y cargue en el campo '*Maximo_promedio'*, el máximo promedio de cada competidor sobre los meses procesados. ***Explicación:*** *observando el dibujo de la matriz, cada competidor tiene sus tiempos cargados mes a mes. El objetivo de este punto es buscar el máximo por fila. Como cada fila de la matriz es manejada con una variable índice que se corresponde con el índice del array de competidores, una vez encontrado el máximo, se guarda en el campo 'Maximo_promedio' del array de competidores usando ese índice. Con esta acción completamos el último campo vacío.*

7. Función que reciba el array de competidores, el array de enteros y cargue en este array**, las posiciones de los registros** del array de competidores, cuyo '*Promedio_de_tiempos_del_mes',* **no supere** al '*Máximo_Promedio'* . Los datos deben almacenarse en forma contigua en el vector de posiciones, por lo tanto tendrá un índice propio. Esta función debe ***retornar*** la cantidad de competidores detectada. ***Explicación:*** *se comparan dos campos del mismo array, hay que recorrerlo y cuando la condición que se pide es verdadera, se guarda la posición del array de competidores en el array de enteros. Este tendrá un índice propio, no se debe utilizar el mismo índice que el array de competidores, ya que el ejercicio pide que se guarde en posiciones contiguas de memoria. Al guardar un índice, debe incrementarse el índice del*

array de enteros para poder ocupar la posición siguiente del mismo. Este índice también indica la cantidad que se está cargando y es el valor que se retornará.

8. Función que reciba el array de competidores, array de instructores, array de posiciones y la cantidad retornada en la función anterior y emita el campo 'P*rovincia'* de los competidores obtenidos en la función anterior y el campo '*Nombre_del instructor'* y *'Nombre_del_competidor'. **Explicación**: la posición de los competidores obtenidos en la función anterior, están guardadas en el array de enteros y además tengo la cantidad, con lo cual, recorriendo el array de competidores hasta la cantidad y colocando como índice el contenido del array de enteros obtengo el campo 'Provincia' solicitado. Pero además me piden nombres del instructor y el del competidor que están en el array de instructores, por tal motivo necesito un ciclo interno que por cada posición del ciclo externo recorra todo el array de instructores y compare el id de ambos para poder emitir estos datos en correspondencia con la provincia.*

9. Función que reciba el array de competidores y lo ordene por **provincia** en forma ascendente. ***Explicación**: se debe comparar la provincia actual con la siguiente, si esta es menor deben intercambiarse los datos. Para eso necesitamos una variable auxiliar de tipo struct competidores, cuando se realiza el intercambio se hace por posición completa, es decir, si los datos estuviesen en forma de tabla, sería la fila completa, NO se debe hacer campo por campo (aunque funcione).*