2do Parcial Laboratorio 1

**LEER ATENTAMENTE Requisitos e instrucciones de entrega:**

1. **Estar conectado a Zoom durante el desarrollo y entrega del parcial, compartiendo pantalla.**
2. **Dentro del main.c, Enumerar los incisos de desarrollo y parte extrema superior escribir: ///APELLIDO Y NOMBRE**
3. **Una vez finalizado, hacer click en File -> export -> PDF -> Seleccionar NO imprimir líneas.**
4. **Subir el PDF a la entrega de tarea del campus. NO SE ACEPTARÁ el main.c ni ningún otro formato.**

**Si no se cumple alguno de estos pasos no se acepta la entrega.**

**El código que aparezca comentado NO se corrige.**

**El parcial tiene una duración de 2hs y 15 minutos, una vez finalizadas no podrán realizar la entrega.**

Enunciado

Se tiene preparada la lista de la selección Argentina que jugará el primer partido para clasificar a cuartos de final en la Copa América. Para llevar una estadística adecuada, se desea trasladar el archivo de jugadores a una estructura adecuada para luego generar las estadísticas asociadas a cada jugador, según haya jugado o no dicho partido.

Nota: Si no recuerdan la función Random, la pueden googlear

Estructura Jugadores

|  |
| --- |
| typedef struct {  char nombre[30];  char tipo[30]; //Portero - Defensa - Mediocampista - Delantero  int nro; // nro que lleva en la camiseta }Jugador; |

1. Leer el archivo “Jugadores.bin” e insertar los datos en un arreglo de Jugadores ordenados por “tipo”.
2. Se desea generar una matriz de 5x5 “*formación*” en donde se indique la formación del equipo en el partido. La idea es que nos llegue la matriz y 5 variables por parámetro a la función, y cargaremos esa cantidad en cada fila según el orden:

EJ: formación 1-4-3-3-0 o 1-3-4-2-1 (orden de llegada de los parámetros) en el cual el resto de la matriz que no posee datos tenga un registro invalido (el nro de camiseta = -1) así identificamos donde finaliza.

La carga es: 1era fila ira el Portero, 2da Fila Defensa, 3era fila Mediocampo, 4ta fila Mediocampo y 5ta fila Delanteros. Por ejemplo, en una formación 1-4-3-2-1, tenemos el portero, 4 defensas, 3 mediocampistas, 2 mediocampistas(serían medias puntas pero simplificamos términos) y 1 delantero.

Quedaria asi:

**Matriz de Registros de tipo Jugador que indica formación de titulares**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Portero | invalido | invalido | invalido | invalido |
| Defensa | Defensa | Defensa | Defensa | invalido |
| Mediocampista | Mediocampista | Mediocampista | invalido | invalido |
| Mediocampista | Mediocampista | invalido | invalido | invalido |
| Delantero | invalido | invalido | invalido | invalido |

La selección de jugadores que debe ir en cada casillero será aleatoria (deben usar random), obviamente correspondiendo al tipo de jugador que debe ir en esa fila (delantero-mediocampista-defensa-portero)

1. A partir de la matriz generada con los jugadores, se juega el partido. Una vez jugado se indican las estadísticas de todos los jugadores (estén en la matriz o no) en el siguiente registro:

|  |
| --- |
| typedef struct {  char nombre[30];  int pj; ///partidos jugados , 0 por defecto  int goles; //0 por defecto }Est\_Jugador; |

La cantidad de goles en el partido es un número random entre 0 y 5 y la selección de qué jugador hizo cada gol será aleatoria (debe estar en la matriz de formación, y no puede ser el portero)

Una vez generado el arreglo de registros de todos los Est\_jugador, se deberá escribir en un archivo de salida denominado *“estadisticas.bin”*

**Tabla de Puntuaciones**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Inciso** | **1** | **2** | **3** | **Invoco correctamente a cada una de las funciones en el main?** | **Nota final** |
| **Puntaje** | **20** | **40** | **40** | **Si/No** |  |
| **Obtenido** |  |  |  |  |  |