

# PARCIAL 2

Cátedra Estructuras de Datos I

13 de Junio de 2017

## 1. Presentación del ejercicio

En el día de ayer nos empezó a sonar el celular, al ir a revisar vemos una solicitud para iniciar una videollamada de un número desconocido; por supuesto que la rechazamos. Al minuto, nos llega un mensaje que dice: Atendeme, soy Sampaoli necesito hablar con vos. No podíamos todavía salir de nuestro asombro cuando nos vuelve a sonar, aceptamos la videollamada y, ¡cuál sería nuestra sorpresa al ver a nuestro nuevo seleccionador nacional con su familia solicitando nuestra colaboración!



Sabrás que estoy revolucionando la Selección, nos dijo, no quiero que eso sólo sea en la mentalidad y el equipo, quiero que eso vaya más allá, quiero patear el tablero: ¡quiero que se dejen de usar listas para usar árboles!



No entendíamos lo que nos quería decir, entonces nos explico, quiero que se deje de pensar en una lista, secuencial, para pensar en árboles. Ahora, yo veo caras de jugadores y me las imagino como nodos de un árbol.

Le pedimos entonces que nos diga qué teníamos que hacer y, nos dijo que era muy simple: la información que tenía almacenada en una lista la quería volcar en un árbol para que la misma estuviera organizada de una forma más eficiente. Nos comentó que un jugador era representado con: país, nombre, apellido, club en el que juega, edad, posición en la que juega (del 1 al 11). Instantáneamente, se nos ocurrió plantear una estructura así:




```
typedef struct {  
    char* nombre, *club;  
    int posicionJugador, edad;  
} _Jugador;  
  
typedef _Jugador* Jugador;  
  
typedef struct nodo{  
    Jugador jug;  
    struct nodo* izquierda, * derecha;  
} Nodo;
```

Luego, preguntamos qué teníamos que hacer con la estructura y nos dijo: precisamos poder agregar un nuevo jugador, considerando que queremos poder usar esta estructura para que, en la medida que se van agregando los jugadores, van quedando organizados por su posición.



Es decir, vamos a querer obtener, todos los jugadores que se encuentren en algún rango de posiciones: ¡por esto es que estamos usando un árbol y no una lista! Este cambio nos va a permitir tener otra mirada que nos permita llevar a la Selección Argentina donde tiene que estar.

Cuando escuchamos esto, pensamos en los siguientes prototipos para las funciones que nos estaba mencionando:



```
Nodo* agregaJugador(Nodo* inicio, Jugador j);  
  
Nodo* recuperaJugadoresPorPosicion(Nodo* inicio,  
                                   int posicionMax, int posicionMin);
```

y, toda esta información la vamos a almacenar en un archivo con extensión .h. También nos indicó que la segunda función podía hacer uso de la primera aun en el caso que no sepamos implementarla.

Amablemente, nuestro interlocutor, nos pidió que generemos una prueba de que funcionara el programa por lo que acordamos que íbamos a tener un main que verifique que:

- el retorno de recuperaJugadoresPorPosicion es NULL cuando inicio es NULL;
- el retorno de recuperaJugadoresPorPosicion es NULL cuando posicionMax es menor que posicionMin;
- se agregan varios jugadores y, verificamos que la función recuperaJugadoresPorPosicion retorna NULL cuando buscamos en un rango al que no pertenecen los jugadores

Luego, nos agradeció por resolver esto a la brevedad y, cortó la comunicación.

## 2. Evaluación

Se pide que escriba un código que pueda implementar las funciones solicitadas respetando los tipos definidos y los prototipos indicados anteriormente.

Se tomará como criterio de aprobación:

1. funcionamiento del programa;
2. claridad y eficiencia del código escrito;