Recomendaciones para hacer el ejercicio 3.

En primer lugar, aparte de otras variables necesarias y pedir al usuario el número de elementos deberéis definir un puntero para reservar dinámicamente el vector de estructuras:

struct Ficha_jugador *v;

Ahora este puntero servirá para hacer la reserva dinámica. Podéis hacerlo de dos maneras distintas. Las dos formas las tenéis perfectamente explicadas en el recurso que aparece en el moodle de teoría en el tema5. El recurso aparece con el título "Gestión dinámica de la memoria", en concreto está en las páginas 31 y 32.

Para los dos casos, las llamadas desde el principal serían:
Para la primera forma: ***v = reservarVector(numeroElementos);***
Para la segunda forma ***reservarVector(&v, numeroElementos);***

Para introducir los datos de un jugador, implementar una función que tenga un prototipo como éste:
struct Ficha_jugador introducirJugador();
En ella, se creará un variable local en la que se introducirán los datos de un jugador y se devolverá dicha variable.

Para introducir los datos de todos los jugadores se puede usar una función con un prototipo como éste:
void introducirVectorJugadores(struct Ficha_jugador *v, int numeroElementos);
En ella, se recorrerán los elementos del vector con un for y a cada elemento se le asignará una llamada a la función ***introducirJugador***(). Poner leyendas que indiquen que numero de jugador se está introduciendo antes de pedirlo, para que le quede claro al usuario.

Para mostrar los datos de todos los jugadores, se puede hacer algo similar a la introducción. Implementar una función que muestre un jugador por pantalla, cuyo prototipo sería:
void mostrarJugador(struct Jugador j);
Implementar otra función que muestre el vector de jugadores, cuyo prototipo sería:
void mostrarVectorJugadores(struct Ficha_jugador *v, int numeroElementos);
En ella, con un for se recorreía el vector y cada vez que se pase por un elemento del vector, se invocaría a la función ***mostrarJugador*** usando como parámetro ese elemento.

Para borrar jugadores con la a en su nombre lo tenemos algo más complicado, aunque se puede hacer de varias formas. Os explico la que a mi juicio creo que es la más simple y en la que además la función de borrado de un elemento de la estructura se puede reutilizar fácilmente. Recordar que uno de los objetivos fundamentales al programar es el implementar código que se pueda reutilizar fácilmente.
La función que sirve para borrar un elemento del vector, en la cual se va a basar la que borre todos los elementos que cumplan una condición, tendrá el siguiente prototipo:
void borrarElementoVectorPorPosicion(struct Ficha_jugador **v, int *numeroElementos, int posicion);
Esta función borrará un elemento del vector conocida su posición (tercer parámetro). Hay que pasarle el puntero por referencia porque el vector que se obtiene al borrar el elemento va a cambiar de tamaño y es posible que la posición de meoria que ocupa no sea la misma. La función hará lo siguiente:
Recorrer todos los elementos del vector desde posicion+1 hasta el último, en este orden y cada vez que se pase por un elemento i, este se copiará en la posición i -1. De esta forma, el hueco que ocupaba el elemento a borrar ha sido eliminado, pero el último elemento del vector está duplicado.

Ahora con la función realloc se redimensiona el vector con un tamaño una unidad menor.
Aquí hay que tener cuidado y reducir el tamaño del vector en una unidad. Fijaros que el número de elementos de pasa por referencia ya que va a ser modificado.
Dentro de esta función hay que tener cuidado al usar el elemento i, y dado que se ha pasado por referencia el vector se usará la siguiente notación (*v)[i]. El paréntesis hay que ponerlo por temas de prioridad de operadores.
Ahora quedaría la función para borrar los jugadores que cumplen la condición. Esta función tendrá el prototipo:
void borrarElementosVector(struct Ficha_jugador **v, int *numeroElementos);
En esta función, mediante un while (no usar for ya que el recorrido irá disminuyendo a medida que borramos jugadores) se va pasando por los elementos del vector y cuando encontremos uno que cumpla la condición de borrado, llamaremos a la función de borrarElementoVectorPorPosicion pasándole la posición que ocupa el elemento a borrar. El número de elementos se ha pasado por referencia porque será modificado en la función que borra un elemento por su posición.

Espero que esto os aclare muchas dudas de este ejercicio.