Clase:AVLTree		Método: add	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un árbol AVL y se le agregan 10 elementos de tipo double, que corresponden al usg del jugador y se compara en preorden con un arreglo de jugadores, que se encuentran agregados de la misma manera.	Se crea un objeto de tipo AVLTree	Se agregan 10 elementos de tipo double al árbol.
2	Al igual que en el caso #1 se crea un árbol AVL, pero en esta ocasión se agregan datos de tipo String, que corresponde al nombre del jugador y se compara en preorden con un arreglo de jugadores, agregados, de tal forma que queden organizados en preorden.	Se crea un objeto de tipo AVLTree	Se agregan 10 elementos de tipo String al árbol

Clase:AVLTree		Método: remove	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un árbol AVL y se le agregan 10 elementos de tipo double, que corresponden al usg del jugador. Finalmente se elimina un elemento antes agregado al árbol y se compara en inorden con un arreglo de jugadores, que contiene 9 jugadores ordenados en inorden.	Se crea un objeto de tipo AVLTree y se le agregan 10 objetos de tipo double	Se elimina un jugador del árbol.
2	Al igual que en el caso #1 se crea un árbol AVL, pero en esta ocasión se agregan datos de tipo String, que corresponde al nombre del jugador, después de ser agregados se elimina un jugador y se compara en preorden con un arreglo de 9 jugadores, agregados de tal forma que queden organizados en preorden.	Se crea un objeto de tipo AVLTree y se le agregan 10 objetos de tipo String	Se elimina un jugador del árbol

Clase:AVLTree		Método: search	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un árbol AVL y se le agregan 10 elementos de tipo double, que corresponden al usg del jugador. Luego se busca un jugador de los que han sido agregados anteriormente.	Se crea un objeto de tipo AVLTree y se le agregan 10 objetos de tipo double	Se encuentra un jugador perteneciente al árbol.
2	Se crea un árbol AVL y se le agregan 10n elementos de tipo double, que corresponden al usg del jugador. Luego se busca un jugador que no ha sido agregado al árbol.	Se crea un objeto de tipo AVLTree y se le agregan 10 objetos de tipo double	Null. No se encuentra el jugador, ya que no está agregado en el árbol.

Clase:AVLTree		Método: clear	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un árbol AVL y se le agregan 10 elementos de tipo double, que corresponden al usg del jugador y luego se hace uso del método clear para eliminar todos los jugadores agregados anteriormente.	Se crea un objeto de tipo AVLTree y se le agregan 10 objetos de tipo double	0, ya que al usar el método clear el tamaño del árbol es igual a 0.

Clase:AVLTree		Método: size	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un árbol AVL y se le agregan 10 elementos de tipo double, que corresponden al usg del jugador. Y se hace uso del método size para conocer el tamaño final del árbol	Se crea un objeto de tipo AVLTree y se le agregan 10 objetos de tipo double	10, ya que se han agregado 10 jugadores anteriormente.

Clase:RedBlackTree		Método: add	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un árbol RedBlackTree y se le agregan 10 elementos de tipo double, que corresponden al usg del jugador y se crea un arreglo, al cual se le agregan jugadores por orden de nivel. Finalmente se comparan entre sí (árbol y arreglo) para conocer si son iguales	Se crea un objeto de tipo RedBlackTree	Se agregan 10 elementos de tipo double al árbol.

Clase:RedBlackTree		Método: remove	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un árbol RedBlackTree y se le agregan 10 elementos de tipo double, que corresponden al usg del jugador. Finalmente se elimina un elemento antes agregado al árbol y se compara por orden de nivel con un arreglo de jugadores, que contiene 9 jugadores ordenados de la misma manera (orden de nivel)	Se crea un objeto de tipo RedBlackTree y se le agregan 10 objetos de tipo double	Se elimina un jugador del árbol.

Clase:RedBlackTree		Método: clear	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un árbol RedBlackTree y se le agregan 10 elementos de tipo double, que corresponden al usg del jugador y luego se hace uso del método clear para eliminar todos los jugadores agregados anteriormente.	Se crea un objeto de tipo RedBlackTree y se le agregan 10 objetos de tipo double	0, ya que al usar el método clear el tamaño del árbol es igual a 0.

Clase:RedBlackTree		Método: size	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un árbol RedBlackTree y se le agregan 10 elementos de tipo double, que corresponden al usg del jugador. Y se hace uso del método size para conocer el tamaño final del árbol	Se crea un objeto de tipo RedBlackTree y se le agregan 10 objetos de tipo double	10, ya que se han agregado 10 jugadores anteriormente.

Clase:RedBlackTree		Método: search	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un árbol RedBlackTree y se le agregan 10 elementos de tipo double, que corresponden al usg del jugador, después de haber agregado los jugadores, se elimina uno de ellos. Luego se busca un jugador de los que han sido agregados anteriormente, pero no el que ha sido eliminado	Se crea un objeto de tipo RedBlackTree, se le agregan 10 objetos de tipo double y se elimina uno de ellos.	Se encuentra un jugador perteneciente al árbol.
2	Se crea un árbol RedBlacktree y se le agregan 10 elementos de tipo double, que corresponden al usg del jugador,después de haber agregado los jugadores, se elimina uno de ellos. Luego se busca el jugador que ha sido eliminado.	Se crea un objeto de tipo RedBlackTree , se le agregan 10 objetos de tipo double y se elimina uno de ellos.	Null. No se encuentra el jugador, ya que ha sido eliminado del árbol.

Clase:BinarySearchTree		Método: add	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un árbol BinarySearchTree y se le agregan 10 elementos de tipo double, que corresponden al usg del jugador y se crea un arreglo, al cual se le agregan jugadores por orden de nivel. Finalmente se comparan entre sí (árbol y arreglo) para conocer si son iguales	Se crea un objeto de tipo BinarySearchTree	Se agregan 10 elementos de tipo double al árbol.

Clase:BinarySearchTree		Método: remove	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un árbol BinarySearchTree y se le agregan 10 elementos de tipo double, que corresponden al usg del jugador. Finalmente se elimina un elemento antes agregado al árbol y se compara por orden de nivel con un arreglo de jugadores, que contiene 9 jugadores ordenados de la misma manera (orden de nivel)	Se crea un objeto de tipo BinarySearchTree y se le agregan 10 objetos de tipo double	Se elimina un jugador del árbol.

Clase:BinarySearchTree		Método: clear	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un árbol BinarySearchTree y se le agregan 10 elementos de tipo double, que corresponden al usg del jugador y luego se hace uso del método clear para eliminar todos los jugadores agregados anteriormente.	Se crea un objeto de tipo BinarySearchTree y se le agregan 10 objetos de tipo double	0, ya que al usar el método clear el tamaño del árbol es igual a 0.

Clase:BinarySearchTree		Método: size	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un árbol BinarySearchTree y se le agregan 10 elementos de tipo double, que corresponden al usg del jugador. Y se hace uso del método size para conocer el tamaño final del árbol	Se crea un objeto de tipo BinarySearchTree y se le agregan 10 objetos de tipo double	10, ya que se han agregado 10 jugadores anteriormente.

Clase:BinarySearchTree		Método: search	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un árbol BinarySearchTree y se le agregan 10 elementos de tipo double, que corresponden al usg del jugador, después de haber agregado los jugadores, se elimina uno de ellos. Luego se busca un jugador de los que han sido agregados anteriormente, pero no el que ha sido eliminado	Se crea un objeto de tipo BinarySearchTree, se le agregan 10 objetos de tipo double y se elimina uno de ellos.	Se encuentra un jugador perteneciente al árbol.
2	Se crea un árbol BinarySearchTree y se le agregan 10 elementos de tipo double, que corresponden al usg del jugador,después de haber agregado los jugadores, se elimina uno de ellos. Luego se busca el jugador que ha sido eliminado.	Se crea un objeto de tipo BinarySearchTree , se le agregan 10 objetos de tipo double y se elimina uno de ellos.	Null. No se encuentra el jugador, ya que ha sido eliminado del árbol.

Clase:FibaApplication		Método: searchByCriteriaAge()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un objeto de tipo FibaApplication y se compara el valor esperado (este valor fue comprobado utilizando la herramienta de Excel)	Se crea un objeto de tipo FibaApplication	Los valores son iguales.

Clase:FibaApplication		Método: searchByCriteriaName()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un objeto de tipo FibaApplication y se compara el valor esperado (este valor fue comprobado utilizando la herramienta de Excel)	Se crea un objeto de tipo FibaApplication	Los valores son iguales.

Clase:FibaApplication		Método: searchByCriteriaDefense()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un objeto de tipo FibaApplication y se compara el valor esperado (este valor fue comprobado utilizando la herramienta de Excel)	Se crea un objeto de tipo FibaApplication	Los valores son iguales.

Clase:FibaApplication		Método: searchByCriteriaDRB()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un objeto de tipo FibaApplication y se compara el valor esperado (este valor fue comprobado utilizando la herramienta de Excel)	Se crea un objeto de tipo FibaApplication	Los valores son iguales.

Clase:FibaApplication		Método: searchByCriteriaHeight()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un objeto de tipo FibaApplication y se compara el valor esperado (este valor fue comprobado utilizando la herramienta de Excel)	Se crea un objeto de tipo FibaApplication	Los valores son iguales.

Clase:FibaApplication		Método: searchByCriteriaOffense()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un objeto de tipo FibaApplication y se compara el valor esperado (este valor fue comprobado utilizando la herramienta de Excel)	Se crea un objeto de tipo FibaApplication	Los valores son iguales.

Clase:FibaApplication		Método: searchByCriteriaTeam()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un objeto de tipo FibaApplication y se compara el valor esperado (este valor fue comprobado utilizando la herramienta de Excel)	Se crea un objeto de tipo FibaApplication	Los valores son iguales.

Clase:FibaApplication		Método: searchByCriteriaUSG()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un objeto de tipo FibaApplication y se compara el valor esperado (este valor fue comprobado utilizando la herramienta de Excel)	Se crea un objeto de tipo FibaApplication	Los valores son iguales.

Clase:FibaApplication		Método: searchByCriteriaAST()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un objeto de tipo FibaApplication y se compara el valor esperado (este valor fue comprobado utilizando la herramienta de Excel)	Se crea un objeto de tipo FibaApplication	Los valores son iguales.

Clase:FibaApplication		Método: searchByCriteriaWeight()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un objeto de tipo FibaApplication y se compara el valor esperado (este valor fue comprobado utilizando la herramienta de Excel)	Se crea un objeto de tipo FibaApplication	Los valores son iguales.

Clase:FibaApplication		Método: searchByCriteriaYear()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	Se crea un objeto de tipo FibaApplication y se compara el valor esperado (este valor fue comprobado utilizando la herramienta de Excel)	Se crea un objeto de tipo FibaApplication	Los valores son iguales.