**Part A --> Linked List**

* I created a Tournament class that is actually my Linked List.
* I have created the Node class which is actually the element or an element of my Linked List
* Whereby the Node has a data point which has a datatype of Team that is where the 3rd class comes into play
* Now we I have created the 3rd class which is the Team class which is more or less of acting as my custom datatype for my Node class.
* So with regards to registering team I have created an insertion at the end method as my helper function so that I more or less of append element at the end of the Linked list.
* With regards to the deregister particular team and insertion before the whole idea behind was getting the index of the team then look for the element less than the required element as far as index goes
* **Example**
* **--> 5 --> 6 --> 7**
* **Say I wanted to remove element which is of index 2 then I would stop at index 1 so that I can be at the index 1 then I will point the ref pointer to null so that I have that index 2 being removed from the Linked list**

**Part B --> Sorting**

* So here I have used merge sort so that can sort out the Linked list based on the constraints that I was given that I have to first check if the final Score is greater than the other.
* If the final score is thus equal then I have to check based on the teamName else then if all fails I have to check based on the teamNumber that is always unique that which we know will always hold somehow somewhere.

**Part C --> Dynamic Sorting**

* So here I still use merge sort but based on what the user wants to sort I.e if the user chooses that they want to sort by final score I use the final score to sort and I have merge sort algorithm backing it.