HEWR TARKHANY

CSCD300 HW2

```
    public void addLast(Object data) {
```

1. This method is for doubly linked list create a new node and with help of the node constructure we get the data next and prev into the new node
2. We assign the NN. Next gets head because next of new node is should be connected to the head from the back
3. And also prev of NN should also be connected to head.preve because that way it will connected the last element to the list to the NN we are adding
4. Then this.heAD.PREV.NEXT will be connected to NN from the front also this.head.prev from the back that we NN will be connected to the list
5. Then size decremented by one

```
    public CDoublyLinkedList subListOfSmallerValues(Comparable data) {
```

1. We check if the size of this list is smaller or equal to zero if yes then we return the list because the list will be empty
2. Then we create a new a LinkedList Object assign to empty linkedlist constructor
3. We create a node reference CUR so we can walk to the this list from the head till end of the list which is head.prev
4. If we find any data that is the smallest then we add them to the new list

   By addlast method

   Walk through the list by cur=cur.next till the end
5. returning the new list that was created

```
public boolean removeStartingAtBack(Object dataToRemove)
```

1. we create cur as reference of this.head.prev also pref as a head right behind cur
2. then we use a while loop to walk the list from the back we say while cur is not equal to the head then walk the list bur pref gets cur and cur = cur.prev
3. if the currenct Data and the data that is past in are null then we will walk and skip the node we want to remove. It means we found it null == null

   we decrement size and return true it means we deleted.

4.also by if statement we check that if current data is not null and it is equal to the data that is past in then we remove it by

Pref.prev = cur.prev

Cur.prev.next=cur.next

Decrement size then return true because we deleted a node we found

In the end we return false if we do not find anything element.

```
public int lastIndexOf(Object o)
```

1. we create a node reference call it cur and assign it to head.next

   also we create and integer to count how many elelemts we gone through while we are walking through the list
2. in a for loop we create and index I assign it to zero to go through the list

   if both data are null then we do the walk till we get the last index of the match

   count value gets I

3. if current data is not null and its equal to the data we past in then it means we still walk and count gets the value of i
4. out side the for loop we return count because count is in the last index

```
public boolean retainAll(CDoublyLinkedList other)
```

1. we check if the other link is empty or null we will throw and exception if yes
2. we create a Boolean variable so we can return it in the end we assign it to fals
3. we create 2 cur and 2 pref each one of them for each list so we can compare the current ones
4. after the while loop we set the cur1 for this list to go to the list till the end of the list
5. then in side the first loop we create the second loop to go to the other list

   we create another Boolean to keep track it how many items we find and will trun to true when we find items

6.if cur.data equals to other.cur.data then return true else keep going by pref2 get cur2 and cur2 get cur2.next

7. then we say if is true then we cut out the element we found by skiping and walking over them cur.next.prev= cur.prev and cur.prev.next = cur.next and return true again

8. in the end we return the Boolean we created if we do not find anything then it returns false

```
public void insertionSort()
```

1. we create 2 node variables sortedlast and sortedwalker and one object variable unsorrted object
2. we go through the loop by a for loop assigning sortedlast to the head.nexy index 0 and if that got to the last index then we walk to the list by sort= sort.next then we assign its data to the unsorted one
3. second for loop to go backward trough the sorted items from the end of the list till the index 0. By   so2rtedwalk1er get sorted and till we get index zero we decrement sortedwalked = sortedwalker.prev

   then we shift to the right if the data of the sortedwalker is bigger than the unsorted by sortedwalker.next.data = sortedwalker