Hewr Tarkhany

Cscd300 hw3

```
private void addLast(Object e)
```

1. I created this method to add a node to the end of the linkedlist in the reverse method
2. First creat1e a new node and assign the head to a walker node as a reference
3. Then I used a while loop to go through the list by creating CUR reference to do the walk
4. When cur.next == null it means we are in the end of the list
5. Then we add the new node in to the end of the list.

```
6. private MyLinkedList reverse(ListNode node)
```

1. pre condition when the node that is past in is null return a new liked list because if the list is null we need to return a new linkedlist object
2. base case is we cut the problem to smaller problems by creating the a sub list starts from the node.next and leave head alone which we make this call recursively.
3. Then we add the last node to the sub by including the data of the node we have as past in parameter

```
4. private ListNode reverse(ListNode first, ListNode second)
```

1. for the second reverse method we have first node and the second which we use
2. we create a new node as a newh1ead
3. then while the second!== null it means we read the list till the end of the list
4. newhead node get the value of the recursively get the vale of second and second.next as the first node.
5. While doing that then second.next gets the first and first.next = null this will make the next of first to not be connected to anything
6. T3hen we return the newhead
7. After the if then we return first.

```
public int countSpace(String str)
```

1. We creat array of string to split the string past in to a null;
2. And also base case would be if the length of the str == zero it means there is no array and we return 0 if there is not array
3. Then if the first array is a space then we make a recursive call from the index 1 to the end and plus 1
4. If not then we just recursively call the method and cut the first index

```
5. public boolean myContains(String s1, String s2)
```

1.we check if s1 or s2 is null or s1 length and s2 is smaller or equal to zero it means the string is null then we return false

2. if the s2 is contains or starts with s1 then return true it means it contains

3. else recusively calling the method s1 and the substring of s2 and from the index 1.

```
public int div(int m, int n)
```

1. if n is zero it means its not gonna device because we can not device anything then we throw and exception
2. or if n is zero or n is bigger than m we will return zero because we do not want negative values
3. also if n-m is equal zero it means they are equal then we return 1
4. then we do the recursive call from m-n as m and then n as it self

```
5. private boolean isSum24(int arr[], int targetSum)
```

1. if the arra length is zero it means there is no array then we return zero
2. if array[0] or first index is equal the target array lengh is one then we returning true it means we found it and all the elements are equal to the target
3. then recursively get the copy range from index one till the end of the array inclusively and for target we negate the target by the first index to reduce the problem to smaller problem

```
4. private void reverseArray(int a[], int low, int high)
```

1. for base case we say if low is bigger or equal to high or array length smaller than 1 it means we do not have a array or array of size zero we return nothing
2. else we make a swap call which a temp integer gets the value of low and array sub low gets array sub high
3. then temp gets high
4. after that we will reclusively call the array with low incremented by one to move it to the next index and high by -1 to move toward the low by 1.

```
5. private void recursiveSelectionSort(int a[], int low, int high)
```

1. if low is bigger than high then return nothing because there will be no array that index zero is bigger than the last index

2. we creat refences for the smallest value to be in arra[low] and also a reference of the low as index
3. then we go to the array with a searcher from low +1 and high inclusively
4. if the searcher or walker is smaller than the value of low the smallest value gets the arra[searcher] and inde1x gets the searcher
5. we do a swap the min index gets the lowest
6. and lowes3t value gets the arra[low]
7. recursively calling the method array from low +1 and the igh