Instructions for students:

1. Complete the following methods on Linked lists.
2. You may use any language to complete the tasks.
3. You need to submit one single file containing all the methods/functions.The form will be open till 2 days after the due date. 30% marks will be deducted for each late day in case of late submission.
4. The submission format MUST be maintained. You need to copy paste all your codes in ONE SINGLE .txt file and upload that. If format is not maintained, whole lab submission will be canceled.
5. If you are using JAVA, you must include the **Tester class** containing the main method which should test your other methods in **MyList class** and print the outputs according to the tasks.
6. If you are using PYTHON, make sure your code has the methods invoked through **test statements** and proper printing statements according to the tasks. You may create a separate **Tester class** for this or test the methods within the **MyList class**.
7. Usage of built in methods/libraries are NOT ALLOWED
8. The google form link for this lab is provided in BUX under LAB 2-Linked Lists subsection under the FALL21 CSE220 lab tab.

Linked List

Task 1:

i) Create a **Node** class which will hold two fields i.e an integer element and a reference to the next **Node**.

ii) Create a **Linked list** Abstract Data Type (ADT) named **MyList**. The elements in the list are **Nodes** consisting of an integer type key (all keys are unique) and a reference to the next node.

[You are not allowed to use any global variable other than head]

Task 2: (Basic operations) 20 marks

1. **Constructor:**

a. MyList (int [] a) or def __init__ (self, a) (5)

   Pre-condition: Array cannot be empty.
   Post-condition:This is the default constructor of MyList class. This constructor creates a list from an array.

2. void showList ( ) or def showList(self) (2)

   Precondition: None.
   Postcondition: Outputs the keys of the elements of the order list. If the list is empty, outputs “Empty list”.

3. boolean isEmpty ( ) or def isEmpty(self) (1)

   Pre-condition: None.
   Post-condition: Returns true if a list is empty. Otherwise, returns false.

4. void clear ( ) or def clear(self) (1)

   Pre-condition: The list is not empty.
   Post-condition: Removes all the elements from a list.

5. void insert (Node newElement) or def insert(self, newElement) (3)
   Pre-condition: None.
   Post-condition: This method inserts newElement at the tail of the list. If an element with the same key as newElement already exists in the list, then it concludes the key already exists and does not insert the key.

6. void insert (int newElement, int index) or def insert(self, newElement, index) (4)

   Pre-condition: The list is not empty.
   Post-condition: This method inserts newElement at the given index of the list. If an element with the same key as newElement value already exists in the list, then it concludes the key already exists and does not insert the key. [You must also check the validity of the index].

7. Node remove (int deleteKey) or def remove(self, deletekey) (4)

   Pre-condition: List is not empty.

Post-condition: Removes the element from a list that contains the deleteKey and returns the deleted key value.

Task 3: (Advanced operations) (20 marks)

1. Write a function to find out the even numbers that are present in the list and output another list with those numbers. (3)

Sample Input	Sample Output
1 -> 2 -> 5 -> 3 -> 8	2 -> 8
101 -> 120 -> 25 -> 91-> 87 -> 1	120

2. Write a function to find out if the element is in the list or not. *(3)*

Sample Input	Sample Output
1 -> 2 -> 5 -> 3-> 8 and 7	False
101 -> 120 -> 25 -> 91-> 87 -> 1 and 87	True

3. Write a function to reverse the list. [You are not allowed to create any other list] (3)

Sample Input	Sample Output
1 -> 2 -> 5 -> 3-> 8	8 -> 3 -> 5 -> 2 -> 1

4. Write a function to sort the list. [You are not allowed to create any other list] (3)

Sample Input	Sample Output
1 -> 2 -> 5 -> 3-> 8	1 -> 2 -> 3 -> 5 -> 8

5. Write a function that prints the sum of the values in the list. (4)

Sample Input	Sample Output
1 -> 2 -> 5 -> 3-> 8	19

6. Write a function that rotates the elements of the list k times. [You are not allowed to create any other list]. (4)

Sample Input	Sample Output
3 -> 2 -> 5 -> 1-> 8, left, 2	5 -> 1 -> 8 -> 3 -> 2
3 -> 2 -> 5 -> 1-> 8, right, 2	1 -> 8 -> 3 -> 2 -> 5

Assignment 02

CSE220 (FALL' 21)

In [2]:

```
# Task 1

class Node:
    def __init__(self, v, n):
        self.value=v
        self.next = n

class MyList:
    def __init__(self,a=None):
        self.head=None
        self.tail=None
        for i in a:
            newNode=Node(i,None)
            if self.head==None:
                self.head=newNode
                self.tail=newNode
            else:
                self.tail.next=newNode
                self.tail= newNode

# Task 2

    def showList(self):
        if self.head==None:
            print("Empty list")
            return
        n=self.head
        while n is not None:
            print(n.value)
            n=n.next

    def isEmpty(self):
        if self.head==None:
            print("True")
        else:
            print(False)

    def clear(self):
        while self.head is not None:
            self.head.value=None
            self.head=self.head.next

    def insert(self, newElement):
        n=self.head
        node = Node(newElement, None)
        investigate=False
        while n is not None:
            if n.value==newElement:
                print("newElement already exists in the list")
                investigate=True
            n=n.next
        if investigate==False:
            if self.head==None:
                self.head=node
                self.tail=node
            else:
                self.tail.next = node
                self.tail = node
    def insert2(self,newElement,index):
        node=Node(newElement,None)
        n = self.head
        investigate=False
        while n is not None:
            if n.value==newElement:
                print("newElement already exists in the list")
                investigate=True
            n=n.next
        if investigate==False:
            n=self.head
            if index == 0:
                node.next = self.head
                self.head = node
            else:
                for i in range(index - 1):
                    n = n.next
                temp = n.next
                n.next = node
                node.next = temp

    def remove(self, deletekey):
        count=0
        y=0
        n=self.head
        temp=None
        if n.value==deletekey:
            s=n.value
            n=n.next
            self.head=n
            return s
        n = self.head
        while n is not None:
            count+=1
            if n.value==deletekey:
                temp=n
                y=count
            n=n.next
        if y==0:
            print("Delete Key is not in list")
        else:
            n = self.head
            for i in range(y - 2):
                n = n.next
            s=n.next
            n.next = s.next
            return s.value

# Task 3

    def even(self):
        head=None
        tail=None
        n=self.head
        while n is not None:
            if n.value%2==0:
                new_Node=Node(n.value,None)
                if head==None:
                    head=new_Node
                    tail=new_Node
```

```python
                else:
                    tail.next=new_Node
                    tail=new_Node
            n=n.next
        v=head
        if head==None:
            print("Empty List")
        else:
            while v is not None:
                print(v.value)
                v=v.next

    def find_out(self,newelement):
        n=self.head
        investigate=False
        while n is not None:
            if n.value==newelement:
                investigate=True
                break
            n=n.next
        print(investigate)

    def reverse(self):
        n=self.head
        Box=None
        while n is not None:
            temp=n.next
            n.next=Box
            Box=n
            n=temp
        self.head=Box

    def sort(self):
        n=self.head
        while n.next is not None:
            tail=n.next
            while tail is not None:
                if n.value > tail.value:
                    temp=n.value
                    n.value=tail.value
                    tail.value=temp
                tail=tail.next
            n=n.next

    def sum_value(self):
        total=0
        n=self.head
        while n is not None:
            total=total+n.value
            n=n.next
        print(total)


    # Hasib's Code (rotate)

    def rotate(self, direction, k):
        if direction == "left":
            if k == 0:
                return
            temp1 = self.head
            count = 1
            while count < k and temp1 is not None:
                temp1 = temp1.next
                count += 1
            if temp1 == None:
                return
            new_node = temp1
            while temp1.next != None:
                temp1 = temp1.next
            temp1.next = self.head
            self.head = new_node.next
            new_node.next = None
        else:
            if self.head != None:
                return self.head
            temp2 = self.head
            count = 1
            while temp2.next != None:
                temp2 = temp2.next
                count += 1
            if k > count:
                k = k % count
            k = count - k
            if k == 0 or k == count:
                return self.head
            temp1 = self.head
            count2 = 1
            while count2 < k and temp1 != None:
                temp1 = temp1.next
                count2 += 1
            if temp1 == None:
                return self.head
            new_node = temp1
            temp2.next = self.head
            self.head = new_node.next
            new_node.next = None
            return self.head

#---------------------Tester-----------------------#

## Task 2 ##

a=[13,3,1,6,4,12,4,9]
l1=MyList(a)
l1.showList()
l1.isEmpty()
l1.insert(4)
l1.remove(2)
l1.showList()
l1.insert2(12,0)
l1.showList()

## Task 3 ##

l1.even()
l1.find_out(6)
l1.reverse()
l1.sort()
l1.rotate("left",2)
l1.showList()
l1.sum_value()
```

```
13
3
1
6
4
12
4
9
False
newElement already exists in the list
newElement already exists in the list
Delete Key is not in list
13
3
1
6
4
12
4
9
newElement already exists in the list
13
3
1
6
4
12
4
9
6
4
12
4
True
4
4
6
9
12
13
1
3
52
```

```
In [ ]:
```