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 **DoublyList 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 **DoublyList 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 3 Doubly Linked Lists subsection under the FALL21 CSE220 lab tab.

Dummy Headed Doubly Circular Linked List

Task 1:

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

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

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

Task 2: (Basic operations) (20 marks)

1. **Constructors: (3)**

a. DoublyList (int [] a) or def __intit__(self,a)

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

2. void showList ( ) or def showList(self) (1)

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

3. void insert (Node newElement) or def insert(self, newElement) (4)
   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.

4. 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].

5. void remove (int index) or def remove(self, index) (4)

   Pre-condition: The list is not empty.
   Post-condition: This method removes the Node at the given index of the list.[You must also check the validity of the index].

6. int removeKey(int deleteKey) or def removeKey(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.

Assignment 3

CSE220

In [1]:

```
####Task 1

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


####Task 2

class DoublyList:
    def __init__(self,a):
        self.head=Node(None,None,None)
        self.head.next=self.head
        self.head.prev=self.head
        for i in a:
            newNode = Node(i,None,None)
            newNode.next=self.head
            newNode.prev=self.head.prev
            self.head.prev.next=newNode
            self.head.prev=newNode

    def showList(self):
        n=self.head.next
        if self.head==self.head.next:
            print('Empty list')
        else:
            while n is not self.head:
                print(n.value)
                n=n.next

    def insert(self, newElement):
        n=self.head.next
        newNode=Node(newElement,None,None)
        while n is not self.head:
            if n.value == newElement:
                print("key already exists")
                return
            n=n.next
        newNode.next=self.head
        newNode.prev=self.head.prev
        self.head.prev.next=newNode
        self.head.prev=newNode

    def insert_two(self,newElement,index):
        n=self.head.next
        length=0
        while n is not self.head:
            length+=1
            n=n.next
        if index>length or index<0:
            print("Index does not exist")
            return
        n=self.head.next
        count=0
        while n is not self.head:
            count+=1
```

```
            if count==index:
                break
            n=n.next
        pred=n
        node=Node(newElement,None,None)
        node.prev=pred
        node.next=pred.next
        pred.next.prev=node
        pred.next=node

    def remove(self, index):
        n=self.head.next
        length=0
        while n is not self.head:
            length+=1
            n=n.next
        if index>=length or index<0:
            print("Index does not exist")
            return
        n=self.head.next
        count=0
        while n is not self.head:
            count+=1
            if count==index:
                break
            n=n.next
        pred=n
        q=pred.next
        pred.next=q.next
        q.next.prev=pred
        q.next=None
        q.prev=None
        q=None

    def removeKey(self,deletekey):
        n=self.head.next
        check=False
        newNode=Node(deletekey,None,None)
        while n is not self.head:
            if n.value==deletekey:
                check=True
                temp=n.prev
                n.next.prev = temp
                temp.next = n.next
                return n.value
            n=n.next
        if check==False:
            print("The deleted key value does not exists")

##########################################################Tester Class##############################

a = [10,20,30,40]
l1 =  DoublyList(a)
#l1.insert(60)
#l1.showList()
#l1.insert_two(21,3)
l1.remove(0)
#l1.removeKey(10)
l1.showList()
```

```
20
30
40
```

In [ ]:

```
```