Instructions for students:

1. Complete the following methods on Linked lists.
2. You may use any language to complete the tasks.
3. 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**.
4. Usage of built in methods/libraries are NOT ALLOWED
5. The google form link for this lab is provided in BUX under LAB 2-Linked Lists subsection under the FALL20 CSE220 lab tab. The submission deadline for this lab assignment is 16/11/2020

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 class variable other than head.]

Task 2: (Basic operations)

1. **Constructors:**

   a. MyList ( )

      Pre-condition: None.

      Post-condition: This is the default constructor of MyList class. This constructor creates an empty list.

   b. MyList (int [] a) or Myst(a)

      Pre-condition: Array cannot be empty.

      Post-condition: This is the default constructor of MyList class. This constructor creates a list from an array.

   c. MyList (MyList a) or MyList(a)

      Pre-condition: List cannot be empty.

      Post-condition: This is the default constructor of MyList class. This constructor creates a list from another list.

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

   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)

   Pre-condition: None.

   Post-condition: Returns true if a list is empty. Otherwise, returns false.

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

   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)

   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)

   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)

   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)

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

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.

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

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

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]

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.

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].

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