Objective(s) : To understand the basic implementation of a singly linked list.

Under folder pack3_ArrayLinkedList. Implement MyLinkedList.Java, inside package code including all the method mentioned in lecture.

task 1: Implement the following methods

public void add(int d) –> add a node with value d at the head of the linked list

public void insert(int d) –> insert into an ordered linked list (ascendingly)

public int find(int d) –> return the index of the node valued d, or -1 if not found.

public void delete(int d) –> delete from a linked list

Note that It's common to have a clearly defined method append(int d) for keeping natural input order to the linked list.

task 2: Implement the following methods

private int size() –> number of elements in the list

private void add(int [] d) -> add to the list with values from d[length – 1] to d[0] i.e. reverse the order from array d because add(int d) inserts d[i] to the front which makes its content reversed order from the input

private void insert(int [] d) -> add to the list with values from d[i]. Since insert always result in ordered list, simply call insert(d[i])

task 3: Implement the following methods

public void q1_rotate_clockwise(int k) -> Rotate the linked list counter-clockwise by k nodes where k is a positive integer not larger than the list's size.

public void q2_reverse() -> Reverse the list's element.

public void q3_remove_dup() -> Remove duplicates (node with duplicate values) from the list (if exists).

public void q4_increment_digits() -> Given a number represented in a linked list such that each digit corresponds to a node in a linked list. Add 1 to it. For example, 1999 is represented as (1->9->9->9) and adding 1 to it should result in (2->0->0->0)

public boolean q5_isPalindrome() -> Given a singly linked list of integers, the method returns true if the list is palindrome, else false.

submission: MyLinkedList_XXYYYY.Java where XX is the first 2 digit and YYYY is the last 4 digit of your student id.

```java
static void demo1() {
    MyLinkedList list = new MyLinkedList();
    list.add(d:5);
    list.add(d:1);
    list.insert(d:4);
    list.insert(d:3);
    System.out.println(list);
    list.delete(d:2);
    System.out.println("5 is at " + list.find(d:5));
    System.out.println(list);
}
static void demo2() {
    MyLinkedList mList = new MyLinkedList();
    mList.insert(d:50);
    mList.insert(d:40);
    mList.insert(d:30);
    mList.insert(d:20);
    mList.insert(d:10);
    System.out.println(mList);
}
```

```
-demo1--------
head->(1)->(3)->(4)->(5)->null
5 is at 3
head->(1)->(3)->(4)->(5)->null
-demo2--------
head->(10)->(20)->(30)->(40)->(50)->null
```

```java
static void q1() {
    int [] d = {10,20,30,40,50};
    MyLinkedList mList = new MyLinkedList();
    mList.insert(d);
    System.out.println("before -> " + mList);
    mList.q1_rotate_clockwise(k:1);
    System.out.println("(k = " + 1 + ")  -> " + mList);
    mList.q1_rotate_clockwise(k:3);
    System.out.println("(k = " + 3 + ")  -> " +mList);
    mList.q1_rotate_clockwise(k:7);
    System.out.println("(k = " + 7 + ")  -> " +mList);
}
static void q2() {
    int [] d = {1,2,3,4,5,6,7,8};
    MyLinkedList mList = new MyLinkedList();
    mList.insert(d);
    System.out.println("before -> " + mList);
    mList.q2_reverse();
    System.out.println("after  -> " + mList);
}
```

```
-q1--------
before -> head->(10)->(20)->(30)->(40)->(50)->null
(k = 1)  -> head->(20)->(30)->(40)->(50)->(10)->null
(k = 3)  -> head->(50)->(10)->(20)->(30)->(40)->null
(k = 7)  -> head->(50)->(10)->(20)->(30)->(40)->null
-q2--------
before -> head->(1)->(2)->(3)->(4)->(5)->(6)->(7)->(8)->null
after  -> head->(8)->(7)->(6)->(5)->(4)->(3)->(2)->(1)->null
```

```java
static void q3() {
    int [] d = {13, 11, 4, 15, 4};
    MyLinkedList mList = new MyLinkedList();
    mList.insert(d);
    System.out.println("before -> " + mList);
    mList.q3_remove_dup();
    System.out.println("after  -> " + mList);
    int [] e = {13, 11, 15, 4};
    mList = new MyLinkedList();
    mList.insert(e);
    System.out.println("before -> " + mList);
    mList.q3_remove_dup();
    System.out.println("after  -> " + mList);
}
```

```
-q3--------
before -> head->(4)->(4)->(11)->(13)->(15)->null
after  -> head->(4)->(11)->(13)->(15)->null
before -> head->(4)->(11)->(13)->(15)->null
after  -> head->(4)->(11)->(13)->(15)->null
```

```java
static void q4() {
    int [] d = {1, 9, 9, 9};
    MyLinkedList mList = new MyLinkedList();
    //mList.add(d);
    mList.insert(d);
    System.out.println("before -> " + mList);
    mList.q4_increment_digits();
    System.out.println("after  -> " + mList);
}
static void q5() {
    int [] d = {21, 33, 33, 21};
    boolean isPalind;
    MyLinkedList mList = new MyLinkedList();
    mList.add(d);
    isPalind = mList.q5_isPalindrome();
    System.out.println(mList + " isPalindrome() = " + isPalind);
    int [] e = {21, 33, 44, 33, 21};
    mList = new MyLinkedList();
    mList.add(e);
    isPalind = mList.q5_isPalindrome();
    System.out.println(mList + " isPalindrome() = " + isPalind);
    int [] f = {1, 9, 9, 9};
    mList = new MyLinkedList();
    mList.add(f);
    isPalind = mList.q5_isPalindrome();
    System.out.println(mList + " isPalindrome() = " + isPalind);
}
```

```
-q4--------
before -> head->(1)->(9)->(9)->(9)->null
after  -> head->(2)->(0)->(0)->(0)->null
-q5--------
head->(21)->(33)->(33)->(21)->null isPalindrome() = true
head->(21)->(33)->(44)->(33)->(21)->null isPalindrome() = true
head->(1)->(9)->(9)->(9)->null isPalindrome() = false
```

Due date: TBA