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

Implement MyLinkedList.Java, inside package \Lab04\pack including all the method mentioned in lectures

task 1: Implement the following methods

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

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

public void insert(int *d*) –> Insert *d* into the list after the node whose value is the largest value less than *d* (in ascending order) – <u>not the same logic on the slide!</u>.

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

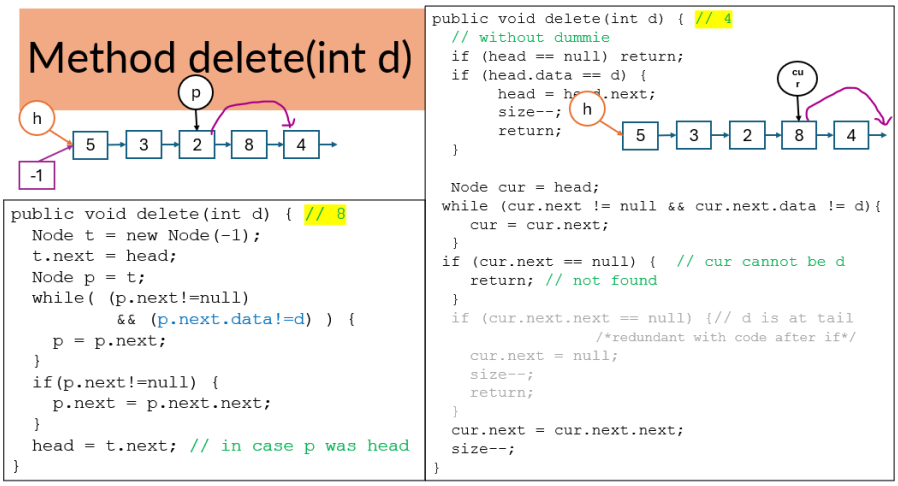
```
static void demo_1() {

    MyLinkedList lis = new MyLinkedList();
    lis.add(5);
    lis.add(1);
    lis.insert(4);
    lis.insert(3);
    System.out.println(lis +
                 " size= " + lis.size());
    lis.delete(2);
    System.out.println("5 is at "
                          + lis.find(5));
    // lis.delete(1); // delete head
    // lis.delete(5); // delete tail
    System.out.println(lis);
}
```

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

public int getAt(int index) *public* void setAt(int index, int d)

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

Hint: Accompanied content may help you completing insert(int d)



task 2: Implement the following methods

public 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

```
static void demo_2() {
    MyLinkedList lis = new MyLinkedList();
    lis.add(new int[] {1,4,5,3});
    System.out.println(lis + " size= "
                            + lis.size());
    lis.delete(3);
    lis.insert(new int[]{8,2});
    System.out.println(lis
                + " size= " + lis.size());
   // [1, 2, 4, 5, 8]
}
```

public 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: Create MyTrickyLinkedList.java. (extends MyLinkedList) Implement the following methods

public void q1_rotate_counter_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 which its value the list already has a node with the value.) from the list (if exists). For simplicity, nodes with the same values are next to each other. Your solution must preserve the order of the values.

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.

```java
static void q1() {
    int [] d = {10,20,30,40,50};
    MyTrickyLinkedList lis = new MyTrickyLinkedList();
    lis.insert(d);
    System.out.println("before -> " + lis);
    lis.q1_rotate_clockwise(4);
    System.out.println("(k= " + 4 + ") -> " + lis);
    lis.q1_rotate_clockwise(7);
    System.out.println("(k= " + 7 + ") -> " + lis);
    lis.q1_rotate_clockwise(1);
    System.out.println("(k= " + 1 + ") -> " + lis);
}
```

```
before -> head->(10)->(20)->(30)->(40)->(50)->null
(k= 4) -> head->(20)->(30)->(40)->(50)->(10)->null
(k= 7) -> head->(20)->(30)->(40)->(50)->(10)->null
(k= 1) -> head->(10)->(20)->(30)->(40)->(50)->null
```

```java
static void q2() {
    int [] d = {1,2,3,4,5,6,7,8};
    MyTrickyLinkedList lis = new MyTrickyLinkedList();
    lis.insert(d);
    System.out.println("before -> " + lis);
    lis.q2_reverse();
    System.out.println( lis );
}
```

```
before -> head->(1)->(2)->(3)->(4)->(5)->(6)->(7)->(8)->null
head->(8)->(7)->(6)->(5)->(4)->(3)->(2)->(1)->null
```

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

```
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};
    MyTrickyLinkedList lis = new MyTrickyLinkedList();
    lis.insert(d);
    System.out.println("before -> " + lis);
    lis.q4_add_one();
    System.out.println("after-> " + lis);
    int [] e = {9,9,9,9};
    lis = new MyTrickyLinkedList();
    lis.insert(e);
    System.out.println("before -> " + lis);
    lis.q4_add_one();
    System.out.println(
      "after-> " + lis);
}
```

```
before -> head->(1)->(9)->(9)->(9)->null
after-> head->(2)->(0)->(0)->(0)->null
before -> head->(9)->(9)->(9)->(9)->null
after-> head->(1)->(0)->(0)->(0)->(0)->null
```

```java
static void q5() {
    boolean isTrue;
    int [] d = {21, 33, 33, 21};
    MyTrickyLinkedList lis = new MyTrickyLinkedList();
    lis.add(d);
    isTrue = lis.q5_isPalindrome();
    System.out.println(lis + " isPalindrome= " + isTrue);
    int [] e = {21,33,44,33,21};
    lis = new MyTrickyLinkedList();
    lis.add(e);
    isTrue = lis.q5_isPalindrome();
    System.out.println(lis + " isPalindrome= " + isTrue);
    int [] f = {1,9,9,9};
    lis = new MyTrickyLinkedList();
    lis.add(f);
    isTrue = lis.q5_isPalindrome();
    System.out.println(lis + " isPalindrome= " + isTrue);
}
```

```
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
```

submission: MyLinkedList_XXYYYY.Java and MyTricklyLinkedList.java.

Due Date: TBA