King Saud University	College of Computer and Information Sciences Department of Computer Science
Data Structures CSC 212	**Practice - Fall 2018**

Question 1 . . . . points

Write the method **public static** `<T>` **void** `removeLast(List<T> l,` **int** `k)`, user of the ADT List, which removes the last `k` elements of `l`. Assume that `l` is **not empty** and that $k >= 0$ and valid (less or equal the list's length).

```
1 public static <T> void removeLast(List<T> l, int k) {
2   ...
3   ...
```

1. Line 2:
   - (A) `l.remove();`
   - (B) `int cpt = 0;`
   - (C) `l.findNext();`
   - [✓ (D)] `int cpt = 1;`
   - (E) None
2. Line 3:
   - (A) `l.findLast();`
   - (B) `l.remove();`
   - [✓ (C)] `l.findFirst();`
   - (D) `l.findNext();`
   - (E) None
3. Line 4:
   - (A) `while (!l.first()){`
   - (B) `while (l.last()){`
   - [✓ (C)] `while (!l.last()){`
   - (D) `for(int i = 0; i < k; i++){`
   - (E) None
4. Line 5:
   - (A) `l.insert(l.retrieve());`
   - [✓ (B)] `cpt++;`
   - (C) `l.insert(k);`
   - (D) `l.remove();`
   - (E) None
5. Line 6:
   - (A) `l.findPrevious(); }`
   - (B) `l.remove(); }`
   - (C) `l.findLast(); }`
   - (D) `l.findFirst(); }`
   - [✓ (E)] **None**
6. Line 7:
   - (A) `l.remove();`
   - (B) `l.findLast();`
   - [✓ (C)] `l.findFirst();`
   - (D) `l.findNext();`
   - (E) None
7. Line 8:
   - (A) `for(int i = 0; i < k; i++)`
   - [✓ (B)] `for(int i = 0; i < cpt - k; i++)`
   - (C) `for(int i = 0; i < cpt; i++)`
   - (D) `for(int i = 0; i <= cpt - k; i++)`
   - (E) None
8. Line 9:
   - (A) `l.findFirst();`
   - [✓ (B)] `l.findNext();`
   - (C) `l.update(null);`
   - (D) `l.remove();`
   - (E) None
9. Line 10:
   - (A) `for(int i = 0; i < cpt; i++)`

(B) `for(int i = 1; i < k; i++)`

[✓ (C) `for(int i = 0; i < k; i++)`]

(D) `for(int i = 0; i < cpt - k; i++)`

(E) None

10. Line 11:

(A) `l.update(null);`

[✓ (B) `l.remove();`]

(C) `l.remove(); l.findNext();`

(D) `l.insert(l.retrieve());`

(E) None

Question 2 . . . . points

Write the method `removeOddElems`, member of the class `LinkedList`, that removes all the elements having an odd position (the position of the first element is 0). Assume that the list is **not empty**. The method signature is: `public void removeOddElems()`.

Example 1. *If $l : A \to B \to C \to D \to E$, then* `l.oddElems()` *returns:* $A \to C \to E$.

```
public void removeOddElems() {
  ...
  ...
```

1. Line 2:

[✓ (A) `Node<T> p = head;`]

(B) `Node<T> p = head.next;`

(C) `Node<T> p = current;`

(D) `Node<T> p = head.next.next;`

(E) None

2. Line 3:

[✓ (A) `while (p != null && p.next != null){`]

(B) `while (p != null){`

(C) `while (p.next.next != null){`

(D) `while (current.next != null){`

(E) None

3. Line 4:

(A) `p.next = p;`

(B) `p = p.next;`

[✓ (C) `p.next = p.next.next;`]

(D) `current.next = p;`

(E) None

4. Line 5:

[✓ (A) `p = p.next;}`]

(B) `p.next.next = current.next;}`

(C) `p = p.next.next;}`

(D) `p.next.next = p.next;}`

(E) None