CSC 212 MT 1 Revision

16/11/2014

Problem 1 (MT1 Fall 2013-Q4.2)

Write the method *removeDuplicates*, member of class *LinkedList*, that removes any duplicates in the list. If an element appears many times, only the first element should be kept, while the duplicates should be removed. Assume that the list is not empty, and use the method *equals* to test for equality. **Do not use any auxiliary data structures and do not call any methods**. The method signature is *public void removeDuplicates()*.

Example 1.1. *If the list contains* $A \to B \to A \to A \to B \to D \to E \to D$*, then after calling* removeDuplicates*, its content becomes* $A \to B \to D \to E$*.*

Solution:

```
public void removeDuplicates() {
        Node<T> cur1 = head;

        while(cur1 != null) {
                Node<T> prev = cur1;
                Node<T> cur2 = cur1.next;
                while(cur2 != null) {
                        if(cur1.data.equals(cur2.data))
                                prev.next = cur2.next;
                        else
                                prev = cur2;

                        cur2 = cur2.next;
                }
                cur1 = cur1.next;
        }
}
```

.....

Problem 2 (HW1-Q3.2)

Write the method *reverse* (member of *LinkedList*), that reverses the list nodes order. **Do not use other class methods or auxiliary data structures**. The method signature is *public void reverse()*.

Example 2.1. *If the list contains:* $A \rightarrow B \rightarrow C \rightarrow D$, *then calling* reverse() *will result in the list becoming:* $D \rightarrow C \rightarrow B \rightarrow A$.

Solution:

```
public void reverse() {
        Node<T> q = null;
        Node<T> p = head;
        while (p != null) {
                Node<T> tmp = p.next;
                p.next = q;
                q = p;
                p = tmp;
        }
        head = q;
}
```

. . . . . .

Problem 3 (HW2-2.1)

	Statement	S/E	Frequency	Total
1	int func(int n) {	0	-	0
2	int sum=0;	1	1	1
3	for(int i=0; i< n^2; i++) {	1	n^2+1	n^2+1
4	for(int j=n-1; j>=n-1-i; j- -) {	1	$n^2(n^2+3)/2$	$n^2(n^2+3)/2$
5	sum=i+j;	1	$n^2(n^2+1)/2$	$n^2(n^2+1)/2$
6	System.out.println(sum);	1	$n^2(n^2+1)/2$	$n^2(n^2+1)/2$
7	}	0	-	0
8	}	0	-	0
9	}	0	-	0
	Total operations			$3n^4/2+7n^2/2+2$
	Big-oh			$O(n^4)$

. . . . . .

Problem 4 (HW3-2.1)

Write the method *intersect*, user of Queue ADT, that accepts two queues q_1 and q_2, and returns the intersection of the two queues as a new queue. There shouldn't be any duplicate elements in the new queue. The elements in the returned queue must have the same order as in q_1. The inputs q_1 and q_2 must not change after the method. The method signature is: *public* $<T>$*Queue* $<T>$*intersect(* *Queue* $<T>q_1$*,* *Queue* $<T>q_2$*).*

Example 4.1.

$$q_1 : B \to A \to C \to D \to E \to G$$
$$q_2 : G \to U \to D \to P \to C$$
$$\textit{Returned queue} : C \to D \to G$$

Solution:

```
public <T> LinkedQueue<T> intersect(LinkedQueue<T> q1, LinkedQueue
   <T> q2) {
        LinkedQueue <T> q3 = new LinkedQueue<T>();
        for(int i = 0; i < q1.length(); i++) {
                T e1 = q1.serve();
                q1.enqueue(e1);
                boolean found= false;
                for(int j= 0; j < q2.length(); j++) {
                        T e2= q2.serve();
                        q2.enqueue(e2);
                        if ((!found) && e1.equals(e2)) {
                                found= true;
                        }
                }
                if (found)
                q3.enqueue(e1);
        }
        return q3;
}
```

.....