CSC 212 Midterm 1 Solution - Fall 2014

College of Computer and Information Sciences, King Saud University
Exam Duration: 2 Hours

18/11/2014

Question 1 [25 points]

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=i; j<i+3; j++) {	1	$4n^2$	$4n^2$
5	sum=i+j;	1	$3n^2$	$3n^2$
6	System.out.println(sum);	1	$3n^2$	$3n^2$
7	}	0	-	0
8	}	0	-	0
9	}			
Total operations			$11n^2+2$	
Big-oh			$O(n^2)$	

2.

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

Question 2 [25 points]

1. See slides.

2.
```
public void updateLast(T e) {
        Node<T> tmp = head;
        while(tmp.next != null)
                tmp = tmp.next;
        tmp.data = e;
}
```

Question 3 [25 points]

1.
```
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;
        }
        tail = head;
        head = q;
}
```

2.
```
public void exchange(int i, int j) {
        T tmp = data[(head + i) % maxSize];
        data[(head + i) % maxSize] = data[(head + j) % maxSize];
        data[(head + j) % maxSize] = tmp;
}
```

Question 4 [25 points]

1.
```
public static <T> void insertList(List<T> l1, List<T> l2, int i){
        l1.findFirst();
        l2.findFirst();
        for(int j = 0; j < i; j++)
                l1.findNext();
        while(!l2.last()){
                l1.insert(l2.retrieve());
                l2.findNext();
        }
        l1.insert(l2.retrieve());
}
```

2.
```
public void aggregate() {
        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 !=
                            cur1)) {
                                // Save the node cur2
                                Node<T> tmp = cur2;
```

```
                                    // Remove cur2
                                    prev.next = cur2.next;
                                    cur2 = cur2.next;
                                    // Put it after cur1
                                    tmp.next = cur1.next;
                                    cur1.next = tmp;
                                    cur1 = tmp;
                            } else {
                                    prev = cur2;
                                    cur2 = cur2.next;
                            }
                    }
                    cur1 = cur1.next;
            }
}
```

Another solution:

```
public void aggregate() {
        Node<T> temp = head.next, insertAfter = head, preTemp = head;
        while (insertAfter.next != null) {
                while (temp != null) {
                        if (insertAfter.data.equals(temp.data) &&
                           temp != insertAfter.next){
                                preTemp.next = temp.next;
                                temp.next = insertAfter.next;
                                insertAfter.next = temp;
                                temp = preTemp.next;
                                insertAfter = insertAfter.next;
                        } else {
                                temp = temp.next;
                                preTemp = preTemp.next;
                        }
                }
                insertAfter = insertAfter.next;
                preTemp = insertAfter;
                temp = insertAfter.next;
        }
}
```