CSC 212 Midterm 1 - Spring 2015

College of Computer and Information Sciences, King Saud University
Exam Duration: 90 Minutes

12/03/2015

Remark: All answers must be written on the answer sheet.

Question 1 [32 points]

1. Complete the linked implementation of *List*. Write down the methods: *public T retrieve(), public void update(T e), public void findNext(),* and *public void insert(T e).*

2. As an implementer, write the method *contains* member of the class *LinkedList*. The method accepts another linked list l_2 as a parameter. It returns *true* when the elements of l_2 are all contained within the linked list in the same order, otherwise, it returns *false*. Assume that both lists are not empty. **Do not use any methods or auxiliary data structures**. The method signature is *public boolean contains(LinkedList<T>l_2)*

   Example 1.1. *Assuming the linked list l:* $1 \rightarrow 2 \rightarrow 5 \rightarrow 3 \rightarrow 10 \rightarrow 6$. *If* l_2*:* $5 \rightarrow 3 \rightarrow 10$, *calling* l.contains($l_2$) *returns true. Similarly, if* l_2*:* $10 \rightarrow 6$, *calling* l.contains(l_2) *returns true. However, if* l_2*:* $2 \rightarrow 3 \rightarrow 10$, *calling* l.contains($l_2$) *returns false. Similarly, if* l_2*:* $1 \rightarrow 2 \rightarrow 8$, *calling* l.contains($l_2$) *returns false.*

Question 2 [32 points]

1. (a) Find the simplest $g(n)$, c and n_0 for the following $f(n)$ s.t: $f(n) \leq cg(n), \forall n \geq n_0 : f(n) = 4n^2 + 9$.

   (b) Order the following functions in increasing growth rate: 2^n, 1, $\log n^2$, $nn!$, $n!$, 4^n, n^2, n

   (c) Find the big Oh notation for the following functions:

   i. $3n^2 + 2n^2 \log(n^2) + n^2 \log(n^4)$.

   ii. $3n! + 10n^3 + 2^n$.

   iii. $2^n + 3^n + n^2 \log(n^n)$.

2. Find the number of steps and the corresponding big-oh notation for the following method. **Copy only the line numbers to the answer sheet; Do not copy the code**:

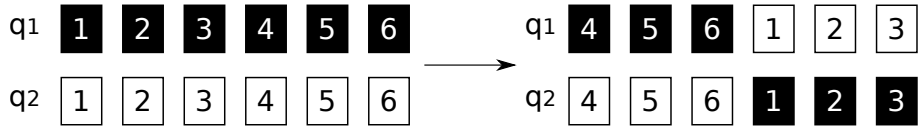
	Statement	S/E	Frequency	Total
1	int func(int n) {			
2	int sum=0;			
3	for(int i=n; i> 0; i- -) {			
4	for(int j=n-1; j>=i; j- -) {			
5	sum=i+j;			
6	System.out.println(sum);			
7	}			
8	}			
9	}			
Total operations				
Big-oh				

Question 3 [36 points]

1. Write the method *public boolean isSymmetric()*, member of the class *DoubleLinkedList* that returns *true* if the list is symmetric about the *current* element, *false* otherwise. Assume that the list is not empty. **Do not use any methods or auxiliary data structures**.

   Example 3.1. *If the list contains:* $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 2 \leftrightarrow 1$ *and the current is on element 3,* isSymmetric *returns true. If it were on any other element it will return false. If the list contains:* $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 4 \leftrightarrow 5$*, then the method returns false for all values of current.*

2. If we have two queues both of an **even same size**, then the crossover of these queues consists in interchanging the halves of the queues as shown in the figure below.



   Write the method *public <T>void crossover(Queue<T>q_1, Queue<T>q_2)*, user of the ADT Queue, that performs a crossover between q_1 and q_2.

Specification of ADT List

- findFirst ( ): **requires**: list L is not empty. **input**: none. **results**: first element set as the current element. **output**: none.
- findNext ( ): **requires**: list L is not empty. Current is not last. **input**: none. **results**: element following the current element is made current. **output**: none.
- retrieve (Type e): **requires**: list L is not empty. **input**: none. **results**: current element is copied into e. **output**: element e.

- update (Type e): **requires**: list L is not empty. **input**: e. **results**: the element e is copied into the current node. **output**: none.
- insert (Type e): **requires**: list L is not full. **input**: e. **results**: a new node containing element e is created and inserted after the current element in the list. The new element e is made the current element. If the list is empty e is also made the head element. **output**: none.
- remove ( ): **requires**: list L is not empty. **input**: none. **results**: the current element is removed. If L is empty, current will point to null. If the next element exists, it is made current, else the first element is made current. **output**: none.
- full (boolean flag): **requires**: none. **input**: none. **results**: if the number of elements in L has reached the maximum then flag is set to true otherwise false. **output**: flag.
- empty (boolean flag): **requires**: none. **input**: none. **results**: if the number of elements in L is zero, then flag is set to true otherwise false. **output**: flag.
- last (boolean flag): **requires**: L is not empty. **input**: none. **results**: if the last element is the current element then flag is set to true otherwise false. **output**: flag.

Specification of ADT Queue

- enqueue (Type e): **requires**: Queue Q is not full. **input**: Type e. **results**: Element e is added to the queue at its tail. **output**: none.
- serve (Type e): **requires**: Queue Q is not empty. **input**: none. **results**: the element at the head of Q is removed and its value assigned to e. **output**: Type e.
- length (int length): **requires**: none. **input**: none. **results**: The number of elements in the Queue Q is returned. **output**: length.
- full (boolean flag): **requires**: none. **input**: none. **results**: If Q is full then flag is set to true, otherwise flag is set to false. **output**: flag.