CMPSCI 187 (FALL 2018) Lab 08: List

The lab is due by 5:00 pm today. Please make sure that you complete your lab assignment in time and submit it to Gradescope by the deadline time.

- Go to ***File -> Make a Copy*** to make an editable copy of this Google Doc for your account
- Follow the instructions below to complete the lab
- When you are done, go to ***File -> Download As -> PDF Document***

Log in to Gradescope and submit your PDF. Remember to submit to **Lab 08**, **NOT Project**!

Section 1: Multiple Choice [3 x 2 points = 6 points]

One correct answer per question. **Select your choice by making that option red and bold.**

1. **In an array-based circular queue, if the front index is 5 and the rear index is 2, which of the following is true?**

(a) The queue has exactly 3 elements.
(b) The queue has exactly 4 elements.
(c) The queue has exactly 6 elements.
(d) The queue has at least 4 elements but may have more.
(e) None of the above is true as we don't know the capacity of the queue.

2. **When implementing an unsorted linked list, what are the Big-O costs of 1) inserting a new element; 2) searching for an element; and 3) removing the element at a given index?**

(a) All O(N)
(b) O(1); O(N); O(N)
(c) O(1); O(N); O(1)
(d) O(N); O(N); O(1)
(e) All O(1).

3. **Which of the following is false in a circular linked queue?**

(a) The cost of enqueue is O(1).
(b) The cost of dequeue is O(1).
(c) The rear node has a link that points to the front node.
(d) Enqueue always modifies the rear pointer in all cases.
(e) Dequeue never modifies the rear pointer in any case.

Section 2: Programming [2 * 10 points = 20 points]

```
class Node<T> {
  public T info;
  public Node<T> link;
  public Node(T in, Node<T> li) { info = in; link = li; }
}
```

Given the above definition of a generic Linked List node (both class variables are public, so you can access them directly without setters and getters), complete the following generic LinkedList class. Specifically, complete the `maxValue` **and** `threshold` **methods.**

- Do **NOT** use iterators or the special for loop (as it's not defined here).
- Do **NOT** create new nodes (i.e. do NOT use the `new` keyword anywhere).
- Do **NOT** use recursion. The code you write must be self-contained without relying on any method or object that isn't shown here.
- If you want to verify that your code works, you can Google '**java online compiler**' to find any online IDE, open and paste **Lab08.java**, fill in your code, try a few test cases to see if it produces the correct result. You should make every effort possible to work out the code first, before testing it with a Java compiler. Remember: during the exam you won't be able to rely on a Java compiler.

```
   public class LinkedList <T extends Comparable<T>> {
  public Node<T> head = null;       // head pointer
```

a) Return the **largest** element you find in the list. If the list is empty, return `null`. If the largest element has duplicates, you can return any of them (for example, the first one in the list). Note that `T` is a generic type that extends the `Comparable<T>` interface (so you should use its **`compareTo`** method to compare two elements of this type). **NO more than 12 lines of code.**

```
   public T maxValue() {
      if (head == null)
          return null;
      T max = head.info;
      for (Node<T> node = head; node != null; node = node.link)
          if (node.info.compareTo(max) > 0)
                max = node.info;
      return max;
   }
   // Continue to the next page
```

b) **Threshold** is an operation that given an element (called threshold), keep only those elements in the list that are **smaller than** the threshold, and remove all other elements (i.e. those that are larger than or equal to the threshold). For example, if the list has `5->8->12->1->10->14->9` to begin with, and the threshold is 10, then after calling threshold method below, the list should contain only `5->8->1>9` because these are all elements that are smaller than the threshold.

Implement this method below. Note that you are **NOT allowed to use the `new` keyword anywhere**. Instead, you must work by rewiring the links of existing nodes. **NO more than 15 lines of code.** Make sure your code handles all cases without causing any `NullPointerException`.

```
public void threshold(T thres) { // thres is the threshold
   for (Node<T> node = head; node != null; node = node.link)
       if (node.link != null && node.link.info.compareTo(thres) >= 0)
             node.link = node.link.link;
   if (head.info.compareTo(thres) >= 0)
             head = head.link;
}
// The end
```