CISC 7124 Midterm Exam

Please complete the exam and email it to Prof. Zhou at nzhou@brooklyn.cuny.edu by midnight on Thursday, March 24. The email should have the subject "CISC 7124 Midterm Exam" and contain at most one attachment file.

Question 1

Write a class, named `OrderedLinkedList`, which has the following specification:

```
import java.util.*;

public class OrderedLinkedList<E extends Comparable<E>> extends LinkedList<E> {
    public OrderedLinkedList()

    public boolean add(E elm)

    public OrderedLinkedList<E> merge(OrderedLinkedList<E> lst)
}
```

An `OrderedLinkedList` object is a linked list whose elements are in non-decreasing order. The constructor `OrderedLinkedList` initializes the list to an empty list. The method `add`, which overrides the one in `LinkedList`, inserts `elm` into the list such that the order is maintained. The method `merge` merges `this` and the given list `lst` into a new ordered list.

Question 2

Implement the following functions as static methods in Java.

1. `public static <T> LinkedList<T> suffix(LinkedList<T> lst, int n)`: This function returns the `n`-element suffix of `lst`. It throws an exception if `lst` has fewer than `n` elements. For example, for `lst = ['a','b','c','d']` and `n = 3`, it returns `['b','c','d']`.

2. `public static <T> T mostFrequent(LinkedList<T> lst)`: This function returns the most frequently occurring element (so called *mode*) of `lst`. For example, the returned value for `lst = [1,3,1,3,2,1]` is 1. If there are multiple most-frequent elements, then the function can return any one of them.

3. `public static int max(int[] arr)`: This function takes an array `arr` that consists of a strictly ascending sequence followed by a strictly descending sequence, and returns the maximum element in `arr` in log_2(`n`) time, where n is the length of the array.

Question 3

An element in a list of integers is called *shaded* if there are elements to its left that are greater than or equal to the element. Write a function that takes a list of integers and returns a copy of the list with all the shaded elements removed.

```
public static ArrayList<Integer> removeShaded(ArrayList<Integer> lst)
```

For example, for `lst = [1,3,3,2,4,3]`, the returned list is `[1,3,4]`.

Question 4

This question continues from Question 3. Suppose a list is represented as a singly linked list of nodes of the `ListNode` class defind below. Write the function of the following specification:

```
public static ListNode<Integer> removeShaded(ListNode<Integer> head)
```

```
class ListNode<T> {
    public T data;
    public ListNode<T> next;

    public ListNode(T element){
        data = element;
        next = null;
    }

    public ListNode(T element, ListNode<T> next){
        data = element;
        this.next = next;
    }
}
```

Question 5

Consider the `BTNode` class:

```
class BTNode<T> {
    T data;
    BTNode<T> left, right;

    public BTNode(T data){
        this.data = data;
        left = null;
        right = null;
    }

    public BTNode(T data, BTNode<T> left, BTNode<T> right){
        this.data = data;
        this.left = left;
        this.right = right;
    }
}
```

Implement the following functions:

1. `depth(BTNode<T> root)`: This function returns the depth of the binary tree under `root`. Assume that the depth of the root is 0, and that the depth of an empty tree is -1.

2. `level(BTNode<T> root, int depth)`: This function returns a list of node values at `depth` in the tree under `root`.

3. `eval(BTNode<Character> root)`: This function returns the value (an integer) of the expression tree under `root`. An expression tree is a binary tree, in which each non-leaf node corresponds to an operator ('+' or '-'), and each leaf node corresponds to an operand ('0', '1', ..., or '9').