Data Structures - Midterm Exam

Please complete the exam and submit it as a plain text email with the subject "CISC 3130 Midterm Exam" to nzhou@brooklyn.cuny.edu by 4PM on Wednesday, March 24.

Question 1:

Consider the function `f()` defined below:

```
static void f(LinkedList<Integer> alist){
    ListIterator<Integer> iter;

    while(alist.size() != 0) {
        iter = alist.listIterator();
        for(int j = 0 ; j < alist.size()-1; j++)
            iter.next();
        System.out.println(iter.next());
        iter.remove();
    }
}
```

Assume `alist` has values `1 2 9 8 10`. What is the output of `f(alist)`?

Question 2:

Consider the following function `f`:

```
static <T> void f(ArrayList<T> v) {
    int i, n;
    n = v.size();
    for (i = 1; i < n; i++){
        v.set(i, v.get(i-1));
    }
    v.remove(0);
    System.out.println(v);
}
```

Assume `v` has values <`1, 2, 3, 4, 5`>. What is the content of `v` after the function call `f(v)`?

Question 3:

Describe the behavior of the function `f` defined below.

```
static <T> void f(T arr[]){
    int n = arr.length;
    LinkedList<T> s = new LinkedList<T>();
    int i;

    for (i = 0; i < n; i++)
        s.add(arr[i]);

    i = 0;
    while (!s.isEmpty()){
        arr[i] = s.poll();
        i++;
    }
}
```

Assume `arr` contains values <`1, 5, 4, 3, 2`>. What is the content of `arr` after the function call `f(arr)`?

Question 4:

The function `createIntArray(n)` creates and returns an `n`×`n` 2-dimensional array of integers, whose elements are:

```
n*n           n*n-1         ...   n*n-n+1
n*(n-1)       n*(n-1)-1     ...   n*(n-1)-n+1
                            ...
n             n-1           ...    1
```

For example, for $n = 3$, the returned array is:

```
{{9, 8, 7},
 {6, 5, 4},
 {3, 2, 1}}
```

Implement the function.

```
int[][] createIntArray(int n);
```

Question 5:

Suppose that the roster of a course is represented as a `LinkedList` of `Student` objects, where the class `Student` is defined as follows:

```
class Student implements Comparable<Student> {
    public String name;
    public String id;
    public float grade;

    Student(String name, String id, float g){
        this.name = name;
        this.id = id;
        grade = g;
    };

    public int compareTo(Student s){
        if (Math.abs(grade-s.grade) < 0.01)
            return 0;
        else if (grade > s.grade)
            return 1;
        else
            return -1;
    }
}
```

Implement a class, named `Roster`, that has the following specification:

```
class Roster {
    LinkedList<Student> students;

    public Roster(){
        students = new LinkedList<>();
    }

    public void addStudent(Student stu);

    public void sort();

    public Pair<float,float> minMax();
}
```

1.1) The method `addStudent` adds a student to the roster.

1.2) The method `sort())` sorts the list of students by grade in descending order.

1.3) The method `Pair<float,float> minMax()` returns a pair whose `first` member is the lowest grade and `second` member is the highest grade of the roster.

Question 6:

The following gives two different implementations for removing duplicates from an array.

```
static <E> removeDups1(ArrayList<E> lst){
    ArrayList<E> res = new ArrayList<>();
    for (E e: lst){
        if (!res.contains(e)){
            res.add(e);
        }
    }
    return res;
}

static <E> removeDups2(ArrayList<E> lst){
    ArrayList<E> res = new ArrayList<>();
    HashSet<E> s = new HashSet<>();
    for (E e: lst){
        if (!s.contains(e)){
            res.add(e);
            s.add(e);
        }
    }
    return res;
}
```

1. What are the best-case and worst-case time complexities of `removeDups1`?
2. What are the best-case and worst-case time complexities of `removeDups2`?
3. Write a function, named `removeConsecutiveDups`, that takes a sorted `ArrayList` and returns a copy of the array with consecutive duplicates removed. For example, for the array `{1,1,2,2,3}`, it returns `{1,2,3}`. Analyze the complexity of your algorithm.

Question 7:

The following function merges three ascendingly sorted lists:

```
static <T extends Comparable<T>>
    LinkedList<T> merge(LinkedList<T> lst1, LinkedList<T> lst2, LinkedList<T> lst3);
```

1. Implement the function.
2. Implement a merge sort algorithm that uses this `merge` function. What are the worst-case and best-case time complexities of your algorithm?