Data Structures - Sample Midterm Exam

Question 1:
Consider the function $f()$ defined below:

```java
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 $7\ 12\ 9\ 8\ 15$. What is the output of $f(alist)$?

Question 2:
Consider the following function $f$:

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

\[
\text{static \(<T>\ \text{void}\ f(T\ arr[])\{}
\text{int}\ n = \text{arr}.\text{length};
\text{Stack<}\ T\\>\ s = \text{new Stack<}\ T\>();
\text{int}\ i;
\text{for (i = 0; i < n; i++)}
\text{s.push(arr[i]);}
\text{i = 0;}
\text{while (!s.isEmpty()){}
\text{arr[i] = s.peek();}
\text{s.pop();}
\text{i++;}
\text{}}
\}
\]

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 \( \text{createIntArray(n)} \) creates and returns an \( n \times n \) 2-dimensional array of integers, whose elements are:

\[
\begin{array}{cccc}
1 & 2 & \ldots & n \\
 n+1 & n+2 & \ldots & 2n \\
 & \ldots \\
 n(n-1)+1 & n(n-1)+2 & \ldots & n*n \\
\end{array}
\]

Implement the function.

\[
\text{int[[]] createIntArray(int n){}
\}
\]
Question 5:

Show the output of the following procedure:

```java
public static void main(String[] args){
    int i;
    Integer arr[] = {6, 3, 3, 4, 9, 6, 9, 3};
    Set<Integer> s = new LinkedHashSet<>(Arrays.asList(arr));
    s.add(10);
    s.add(2);
    System.out.println(s);
}
```

Question 6:

Assume that the roster of CISC 3130 is represented as a LinkedList of `Student` objects, where the class `Student` is defined as follows:

```java
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;
    }
}
```

The member variable `grade` is assumed to hold a real number between 0.0 and 4.0. Implement the following functions:

(a) `static boolean find(LinkedList<Student> roster, float g)`

Returns true if there is at least one student in the roster whose grade is greater than or equal to `g`. 
(b) `void sort(LinkedList<Student> roster)`
    Sorts the list of students by grade in decending order.

(c) `Pair<float, float> low_high(LinkedList<Student> roster)`
    Returns a pair whose first member is the lowest grade and second member is the highest grade of the roster.

**Question 7:**

The following function merges three ascendingly sorted lists:

```java
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 the this function. What are the worst-case and best-case time complexities?