

1 (15). You have the following definitions for LinkedList and Node:

<pre>public class LinkedList {     protected Node head;      public Node getFirst() {         return head;     }      public void setFirst(Node head) {         this.head = head;     } }</pre>	<pre>class Node {     private Object element;     private Node next;      public Node(Object element, Node next) {         this.element = element;         this.next = next;     }      public Object getElement() {         return element;     }      public Node getNext() {         return next;     }      public void setNext(Node next) {         this.next = next;     } }</pre>
---	--

Complete the method below:

```
// This method does a linear search over a linked list and returns the index of the element
// or -1 if the element is not in the list.
```

```
public int doLinearSearch(LinkedList list, Object target) {
```

```
}
```

2. (20) Part A: Sort the following list using Bubble Sort. Show all steps. Is this sort in-place? Why? No code is required.

Index:	0	1	2	3	4	5	6
Value:	9	3	1	8	5	7	3

Part B: Sort the same list using MergeSort. Show all steps. Is this sort in-place? Why?

3 (20). You want to create a robot to neutralize a minefield. For practice, a sample field has been put in a two-dimensional array with width WIDTH and height HEIGHT. The 0s represent unexplored ground, the 1s represent mines, and 2s can be used to represent already explored areas. The starting point will be the top-left corner (0, 0).

```
000000000010
000000000000
000000010000
000000000000
000000000000
000000000000
000000000000
010000000000
000001000000
000000000001
```

Using recursion or a stack, sweep the entire field and print out the locations of all the mines. You only need to write the findMines method and any necessary recursive methods. Remember that the Point class has two public member variables x and y and a constructor of the form Point(int x, int y).

```
public class MineFinder {
    public static final int WIDTH = ###;
    public static final int HEIGHT = ###;

    private int[][] grid;

    ... Methods to initialize field here...

    public void findMines(Point start) {

        }
        // If you choose recursion, treat findMines() as a helper method and write
        // your recursive method(s) on separate page.
    }
}
```

4 (15). You want to remove all duplicate words from a String. Assume that the String has only words and spaces, and has no punctuation. You must use an ArrayList to solve this problem. Remember to parameterize your ArrayList (using bracket notation). Complete the method below.

```
public String removeDuplicates(String words) {
```

```
}
```

5 (15). You have the following recursive method that calculates the nth Fibonacci number:

```
public int fib(int num) {  
    if (num == 1 || num == 2) {  
        return 1;  
    } else {  
        return fib(num - 1) + fib(num - 2);  
    }  
}
```

Convert this into a method that uses a stack instead of making recursive calls.

```
public int fib(int num) {
```

```
}
```

6 (15). Part A. Imagine that you are a tester for a cell phone manufacturing company and you need to test a particular model of cell phone. The phone in question has 9 numeric buttons, a send button and an end button. Write a brief procedure for 3 tests you would perform. Remember to include set-up/tear-down steps where appropriate.

Part B. Specify a pre-condition for the method below. Write a Java *assert* statement to make sure this condition is not violated. Assume that the phone is pre-paid and will stop working if it runs out of minutes.

```
// Use these variables in your method.  
private int numMinutesAvailable;
```

```
// This method deducts minutes from those available after the completion of a call.
```

```
// Precondition 1:
```

```
//
```

```
public void makeDeductions(int numMinutesUsedForCall) {  
    // Your answer here should only be one line long.
```

```
}
```

7 (5). Explain what it means for one thread to join another thread.