XXXday May XX, 2013
XX.XX am/pm – XX.XX am/pm
(Duration: X hour XX minutes)

DEGREES OF MSci, MEng, BEng, BSc, MA and MA (Social Sciences)

Algorithms and Data Structures 2

(Answer all 4 questions.)

This examination paper is worth a total of 50 marks

You must not leave the examination room within the first hour or the last half-hour of the examination. (for exams of 2 hours duration)

or

You must not leave the examination room within the first half hour or the last fifteen minutes of the examination. (for exams of less than 2 hours duration)

1. In the program below method `f` takes as an argument `S`, an array of `String`, and is called in the `main` method. A call to `gen.nextInt(n)`, where n is an integer, delivers a random integer in the range 0 to n-1 inclusive. You should assume that `S` contains no duplicates.

```
import java.util.*;

public class Test {

    static void f(String[] S){
        ArrayList<String> L = new ArrayList<String>();
        int n = S.length;
        Random gen = new Random();
        for (String s : S) L.add(s);
        for (int i=0;i<n;i++){
            int j = gen.nextInt(n-i);
            String s = L.get(j);
            S[i] = s;
            L.remove(s);
        }
    }

    public static void main(String[] args) {
        String[] names = {"ted","alice","poppy","nelson","tarra","rosie"};
        f(names);
        for (String s : names) System.out.println(s);
    }
}
```

(a) Explain what method `f` does and suggest a name for the method. **[4]**

(b) What is the complexity of method `f`? Explain your answer and express the complexity using big-Oh notation. **[4]**

2. A linked list is a data structure consisting of a group of nodes which together represent a sequence. In the class definition below for `List`, the list is implemented as a dynamic data structure of linked nodes, with a `head` node (the head of the list) and an integer `size` (keeping count of the number of nodes in the list).

```
public class Node<E extends Comparable<E>>{

    private E element;
    private Node<E> next;

    public Node(){this(null,null);}

    public Node(E element, Node<E> next){
        this.element = element;
        this.next = next;
    }

    public E getElement(){return element;}

    public void setElement(E element){this.element = element;}

    public Node<E> getNext(){return next;}

    public void setNext(Node<E> next){this.next = next;}

    public boolean equals(Node<E> node){return element.equals(node.getElement());}

}
```

```
public class List<E extends Comparable<E>> {

    private Node<E> head;
    private long size;

    public List(){head = null; size = 0;}

    public Node<E> getHead(){return head;}

    public void setHead(Node<E> node){head = node;}

    public boolean isEmpty(){return head == null;}

    public long size(){return size;}

    public void addFront(E s){...}

    public boolean equals(List<E> L){...}

    public boolean isPresent(E s){...}

    public List<E> intersection(List<E> L){...}

}
```

(a) Give a Java implementation for the undefined methods `addFront`, `equals`, `isPresent` and `intersection`.

- Method `addFront` creates a new `Node`, adds it to the front of the list, updates the head pointer and increments the size of the list. **[2]**
- Method `equals` delivers `true` if the current list (`this`) and the argument `L` contain the same elements in the same order. **[4]**
- Method `isPresent` delivers `true` if there is a node in the current list (`this`) that has an element equal to `s`, `false` otherwise. **[3]**
- Method `intersection` delivers a new list that is the intersection of the elements in the current list (`this`) and the argument `L`. You can assume that neither list contains any duplicate elements. **[3]**

(b) Using big-Oh notation, what is the complexity of `addFront`, `equals`, `isPresent` and `intersection`? **[4]**

3. A binary search tree is a binary tree T such that each internal node v of T stores an item e. Items stored at nodes in the left subtree of v are less than or equal to e, and items stored in the right subtree of T are greater than e.

(a) Insert into an initially empty binary search tree the following items in the order shown: 30, 40, 24, 58, 26, 11, 13, 36. Draw the tree after the insertions have been completed. **[2]**

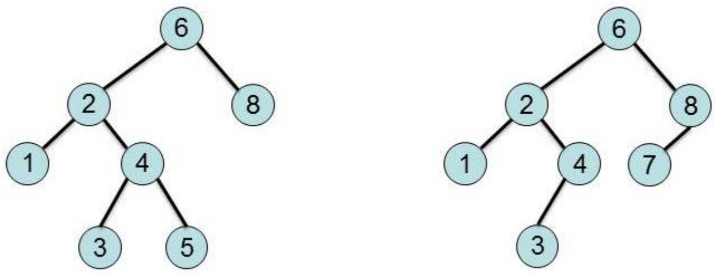
(b) Give the preorder, inorder and postorder traversals of the tree. **[3]**

(c) Draw the tree after the node with item 30 is deleted and describe the algorithm you have used for the deletion. **[3]**

(d) If we had to insert 1023 items into a binary search tree, what could be the minimum and the maximum height of the binary search tree? Suggest what property a data set might have to create maximum height? In your answer explain what we mean by the height of a tree. **[4]**

(e) Briefly describe what is meant by an AVL tree and how an AVL tree avoids the worst case described in part (d) above. **[3]**

(f) Which of the following (if any) are AVL trees? Justify your answer. **[2]**



4. In the class `Sort` below, write java code for the method `locateMax` and `selectionSort`.

(a) The method `locateMax` takes as argument a one dimensional array of integers `S` and an integer upper bound `upb`. The method delivers as a result the location of the largest integer in the array elements `S[0]` to `S[upb]`. **[3]**

(b) Using `locateMax` write java code for the method `selectionSort`, where `selectionSort` takes as argument an array of integers `S` and on termination the integers in `S` are in non-decreasing order. **[3]**

```
public class Sort {

    private static void swap(int[]S,int i,int j){
        int temp = S[i]; S[i] = S[j]; S[j] = temp;
    }

    private static int locateMax(int[]S,int upb){...}

    public static void selectionSort(int[]S){...}

}
```

(c) Prove that the complexity of `selectionSort` is $O(n^2)$. **[3]**