TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

Student ID: .....

EXAMINATIONS — 2008

END YEAR

COMP103
Introduction to
Data Structures and Algorithms

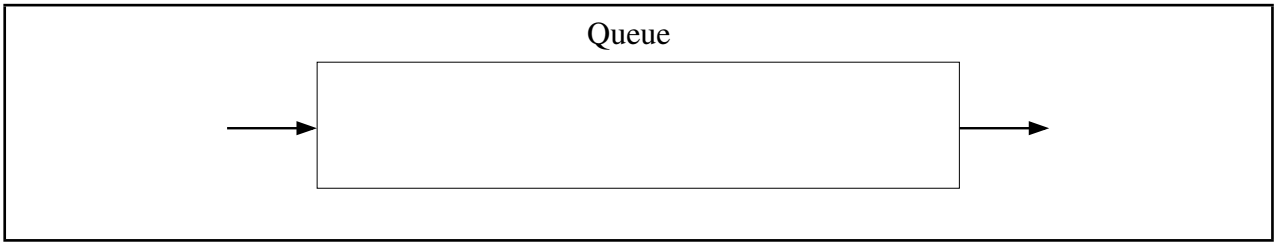
Time Allowed: 3 Hours

Instructions:
1. Attempt **all** of the questions.
2. *Read each question carefully before attempting it.* (Suggestion: You do not have to answer the questions in the order shown. Do the questions you find easiest first.)
3. This examination will be marked out of **180** marks, so allocate approximately 1 minute per mark.
4. Write your answers in the boxes in this test paper and hand in all sheets.
5. Non-electronic translation dictionaries are permitted.
6. Calculators are allowed.
7. Documentation on relevant Java classes and interfaces is at the end of the paper.

Questions	**Marks**
1. Basic Questions	[20]
2. Implementing Collections	[26]
3. Using Collections	[30]
4. Trees	[28]
5. Binary Search Trees	[20]
6. Partially Orderd Trees and Priority Queues	[23]
7. Hashing	[19]
8. Graphs	[14]

Question 1. Basic Questions [20 marks]

(a) [2 marks] Suppose we start with an empty queue and carry out the following operations in order: offer(P), offer(Q), offer(R), poll(), offer(P). Draw the queue after these have been carried out.



(b) [2 marks] What is the *best*-case asymptotic ('big-O') cost of insertion sort?

(c) [2 marks] What is the *average*-case asymptotic cost of selection sort?

(d) [2 marks] Name the general strategy that enables both MergeSort and QuickSort to sort a list efficiently.

(e) [2 marks] How many comparisons are required to find a node in a binary search tree if it contains n items and is perfectly balanced?

(Question 1 continued on next page)

(Question 1 continued)

(f) [2 marks] Which kind of traversal will visit all the values in a Binary Search Tree in their natural order (smallest to largest)?

(g) [2 marks] If you were using a linked list to implement a Stack, would you make the top of the Stack be the first node of the list or the last node of the list? Explain why.

(h) [2 marks] What is a leaf node of a tree?

(i) [2 marks] State the property that must hold for each node in a Partially Ordered Tree.

(j) [2 marks] What is the difference between a directed graph and an undirected graph?

Question 2. Implementing Collections [26 marks]

In lectures we considered the List implementation ArrayList, whose class definition would begin with the declaration of three fields:

```
public class ArrayList <E> extends AbstractList <E> {
    private E[ ] data;
    private int count=0;
    private static final int INITIALCAPACITY=16;
```

(a) [3 marks] It would then continue by giving a constructor - complete the code for this.

```
    public ArrayList () {



    }
```

(b) [12 marks] Complete the code for the following add method for the ArrayList class, which adds the specified element at the specified index. Remember to

- check whether the index is sensible: if it isn't, throw an IndexOutOfBoundsException exception.
- make appropriate use of (but don't write!) the ensureCapacity() method that ArrayList has.

```
    public void add(int index, E item) {

    }
```

(Question 2 continued on next page)

(Question 2 continued)

(c) [3 marks] Lists in java have a method called addAll, which adds all the items from another collection to a list.

addAll is required by the *List* interface, but it is *not* necessary to provide an implementation for it in the *ArrayList* class definition. Explain why not.

(d) [4 marks] Write a version of addAll for the ArrayList class that adds all the items in a collection to the list.

```
public void addAll(Collection<E> other) {




}
```

(e) [4 marks] An efficient version of addAll for an arrayList would only expand the data array once, to be large enough to hold all the items from the collection. Explain why the efficient version would be considerably faster than a simple version that repeatedly called the add method.

Question 3. Using Collections [30 marks]

(a) [10 marks] Complete the code for the following reverseNums method, which is passed a scanner to a file containing only integers. The method uses a stack and returns a list consisting of the integers from the file but in the reverse order.

You will need to create a stack, use the scanner to get the ints, and then build up a list to be returned.

```
public List <Integer> reverseNums(Scanner sc) {




}
```

(Question 3 continued on next page)

(Question 3 continued)

The rest of this question requires you to complete two methods for processing information about hotel bookings in Wellington.

Suppose you are given a file listing the hotels followed by strings indicating the different rooms they have available. For example:

```
HILTON  rm1  rm2   cheapo8 deluxe presidentialA   presidentialB
DUXTON  dux1 dux2  dux3    dux4   dux5  new1  new2
DAYTON  da1  da2
HOLIDAYINN   holA  holB    holC   holD  holE  holF  holG  holH
```

Assume that no two rooms are given the same name.

(b) [8 marks] Complete the readRoomsAndHotels method below, which is passed a Scanner to the above data file, and constructs a Map with rooms as the keys and hotel names as the values in the roomHotelMap field.

```
private Map <String, String> roomHotelMap;
```

Your method should initialise the roomHotelMap field appropriately.

```
public void readRoomsAndHotels(Scanner sc) {









}
```

(Question 3 continued on next page)

(Question 3 continued)

(c) [12 marks] A second data file gives the bookings of hotel rooms around Wellington on a given day. Each line has a room and the name of the person who has booked it:

```
rm1 Bob Smith
dux4 Mary Jones
cheapo8 Helen Clarke
presidentialA Bob Smith
holB Barack Obama
holE John Key
da2 John McCain
```

From this file and the information in roomHotelMap, we would like to find the names of people who have booked rooms in each hotel.

Complete the hotelsToGuests() method on the facing page that is passed a Scanner to a bookings file (as above) and then prints out the set of all the people booked in each hotel. Note that a person might book more than one room in a hotel, but should only be listed at most once for each hotel.

For example, given the bookings file above, the method should print out

```
DUXTON       Mary Jones,
DAYTON       John McCain,
HOLIDAYINN   John Key, Barack Obama,
HILTON       Bob Smith, Helen Clarke,
```

Hint, you may want to construct a Map, with hotels as the keys, and sets of people as the values.

(Question 3 continued on next page)

(Question 3 continued)

```
public void hotelsToGuests(Scanner sc) {

}
```

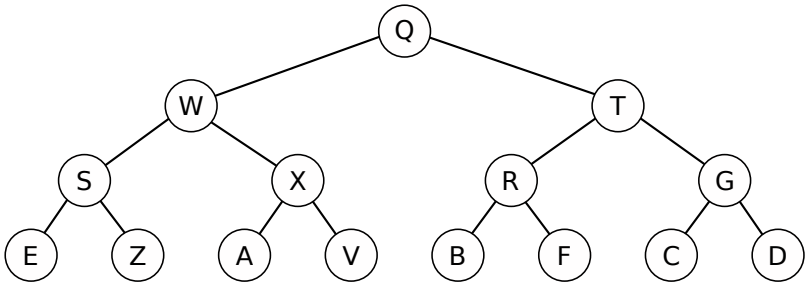
Question 4. Trees [28 marks]

(a) [2 marks] Draw a tree with 9 nodes that is NOT a binary tree and has four levels of nodes.

(Question 4 continued on next page)

(Question 4 continued)

(b) [9 marks] Consider the following (binary) tree.



Show the order that the nodes would be visited by a

(i) [3 marks] Pre-order Depth First Traversal

(ii) [3 marks] in-order Depth First Traversal

(iii) [3 marks] Breadth First Traversal

(c) [3 marks] If a binary tree has n nodes, and every node either has two children or no children, how many leaves are there in the tree? (Hint: you may want to draw some examples).

(Question 4 continued on next page)

(Question 4 continued)

Consider the following TreeNode class that defines nodes for a general tree. Each node has a list of children. The class provides a constructor and two methods: getValue, and getChildren.

```
public class TreeNode <E> {

    private E value;
    private List<TreeNode<E>> children;

    public TreeNode(E v){
        value = v;
        children = new ArrayList<TreeNode<E>>();
    }
    public E getValue(){
        return value;
    }
    public List<TreeNode<E>> getChildren(){
        return children;
    }
}
```

(d) [4 marks] Suppose a variable myTree contains a TreeNode that is the root of a large tree where the root of the tree has at least four children. Complete the following java statement that would print out the value in the third child of the root node.

```
System.out.println(
```

(e) [6 marks] Complete the following leftmost method that will return the value in the leftmost node of a tree.

```
public E leftmost(TreeNode<E> tree){









}
```

(Question 4 continued on next page)

(Question 4 continued)

(f) [4 marks] Complete the following printPostOrderDFT method that will print out the values in the nodes of a tree using a post-order depth first traversal.

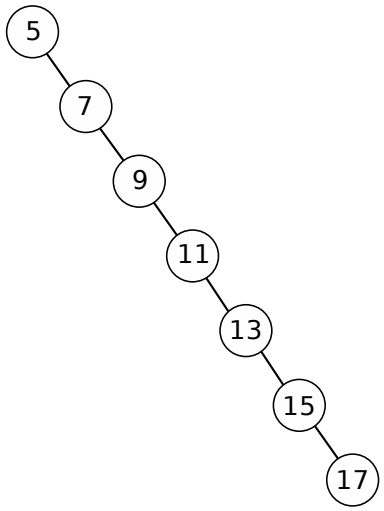
Hint: use a recursive method, not an iterative method.

```
public void printPostOrderDFT(TreeNode<E> tree){




}
```

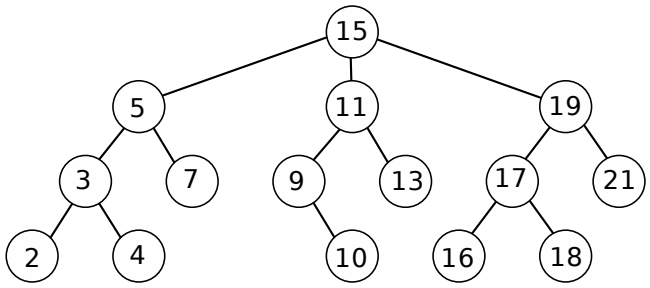
Question 5. Binary Search Trees [20 marks]

(a) [10 marks] For each of the following trees, say whether they are Binary Search Trees or not. If a tree is not, explain why not. The nodes contain integer values.

Tree (i):



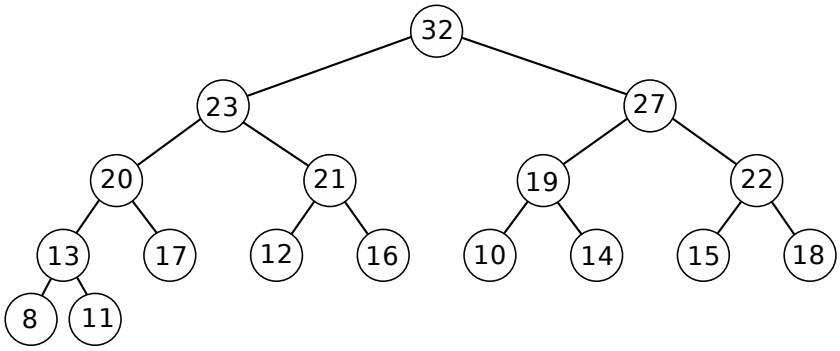
Tree (ii):



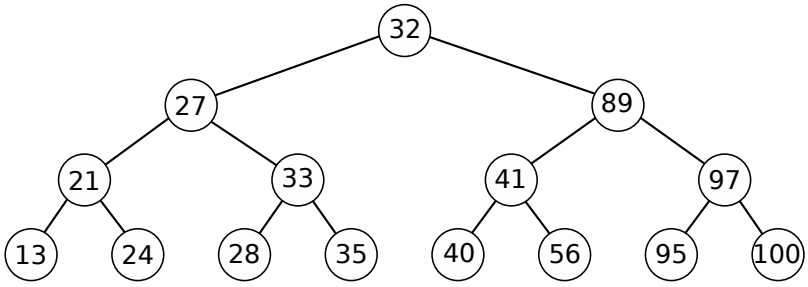
(Question 5 continued on next page)

(Question 5 continued)

Tree (iii):



Tree (iv):



(Question 5 continued on next page)

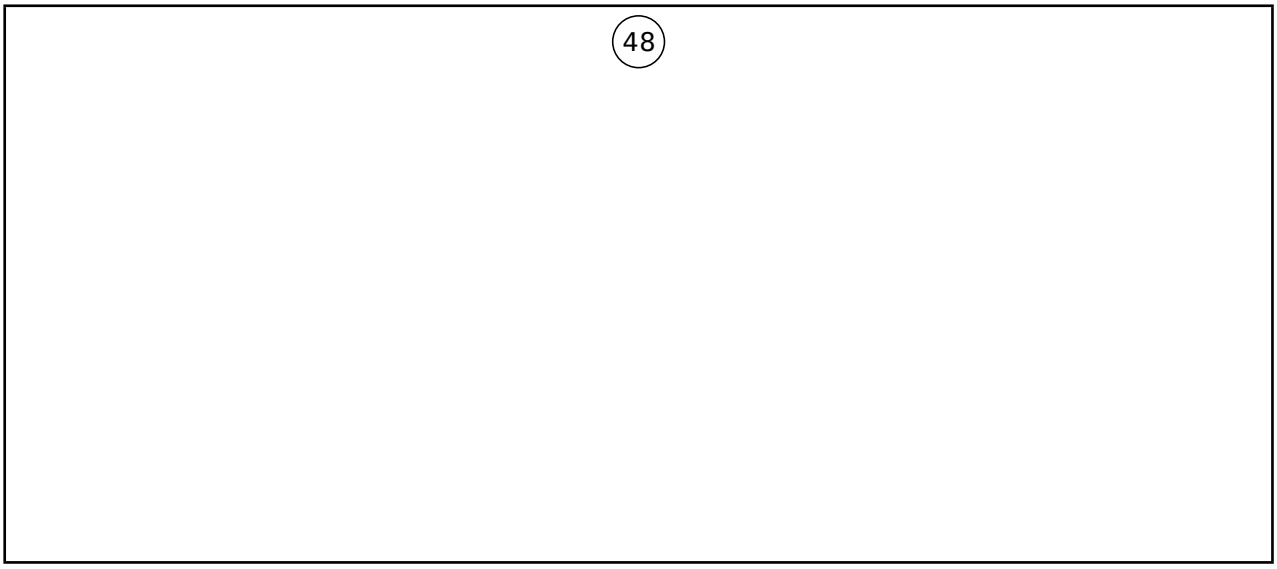
SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

(Question 5 continued)

(b) [4 marks] Show the tree that would be generated by inserting the following values (in order) into the Binary Search Tree below.

72, 71, 23, 47, 49, 70, 25, 18



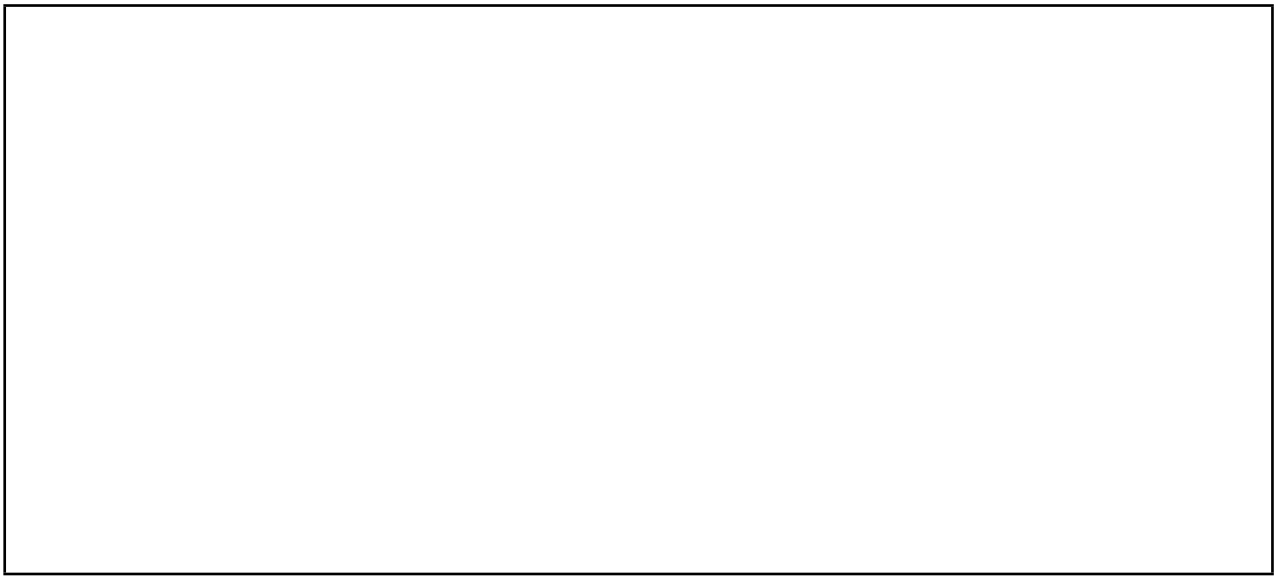
(c) [3 marks] What is the average asymptotic ("Big-O") cost of inserting an item into a well balanced Binary Search Tree containing n nodes? Briefly justify your answer.

(d) [3 marks] What is the least number of comparisons that could possibly be required to insert an item into a BST tree that contains n nodes? Justify your answer by giving an example of a tree and an item leading to this case.

Question 6. Partially Ordered Trees and Priority Queues [23 marks]

(a) [3 marks] Draw a balanced, partially ordered, binary tree containing the following values:

13 18 45 14 27 38 10 31



(Question 6 continued on next page)

(Question 6 continued)

(b) [6 marks] A heap is a partially ordered binary tree implemented in an array. Suppose a heap contains the following 13 values:

98	75	34	61	54	30	31	20	13	45	26	29
0	1	2	3	4	5	6	7	8	9	10	11

Show the sequence of changes to fix up the heap if the largest item (98) is removed from the heap. Each step should show the result after one swap. You may not need all the steps given.

step 1:

0	1	2	3	4	5	6	7	8	9	10	11

step 2:

0	1	2	3	4	5	6	7	8	9	10	11

step 3:

0	1	2	3	4	5	6	7	8	9	10	11

step 4:

0	1	2	3	4	5	6	7	8	9	10	11

step 5:

0	1	2	3	4	5	6	7	8	9	10	11

step 6:

0	1	2	3	4	5	6	7	8	9	10	11

(c) [4 marks] Suppose you need a Priority Queue for items whose priority can be any `double` value. Explain why a heap is a better data structure for implementing the Priority Queue than a sorted array or an array of ordinary Queues (one Queue for each priority).

(Question 6 continued on next page)

(Question 6 continued)

(d) [10 marks] The POTree class below represents a Partially Ordered Tree using a binary tree structure. It contains one field for the root of the tree and a private class for the nodes:

```
public class POTree{

    private POTNode root;

    private class POTNode {
        public int value;
        public POTNode left;
        public POTNode right;

        public POTNode(int v){
            value = v;
        }
    }

  :
```

Write a method called check for the POTree class that checks that the tree in the root field is a proper Partially Ordered Tree. check should return true if the tree is OK, and false otherwise. You may define helper methods if you wish.

Hint: do a recursive traversal of the tree, checking each node.

```
public boolean check(){




}
```

Question 7. Hashing [19 marks]

(a) [4 marks] When two items hash to the same index of the data array, a Hash Set must resolve the collision in some way. Explain the difference between resolving the collision by probing and by using buckets.

(b) [4 marks] A Hash Set with any kind of probing builds up "runs", but quadratic probing is generally better than linear probing. Explain why quadratic probing is better than linear probing.

(c) [3 marks] Why is it important to ensure that a Hash Set using probing is not allowed to get too full?

(Question 7 continued on next page)

(Question 7 continued)

(d) [8 marks] Suppose you are writing a program that needs to store a set of Person objects, and you have decided to use a HashSet. The fields of a Person object are shown below. The important information that identifies an individual is in the name, dateOfBirth, countryOfBirth, and birthCertificateNumber fields.

```
public class Person{
    private final String name;
    private final Date dateOfBirth;
    private final String countryOfBirth;
    private final long birthCertificateNumber;

    private String currentAddress;      // may change
    private long phoneNumber;           // may change
    private String citizenship ;        // may change

      :
```

The other fields in a Person object (like currentAddress) may change over time without changing the identity of the person. The hashCode of a Person object should always be the same, even when these other fields change. Also, two Person objects that have the same values in the identifying fields should be considered to be the same person, and should therefore have the same hashCode (and be equal).

Complete the following hashCode and equals methods for the Person class. You may assume that the String and Date classes have appropriate hashCode functions.

```
    public int hashCode(){




    }
```

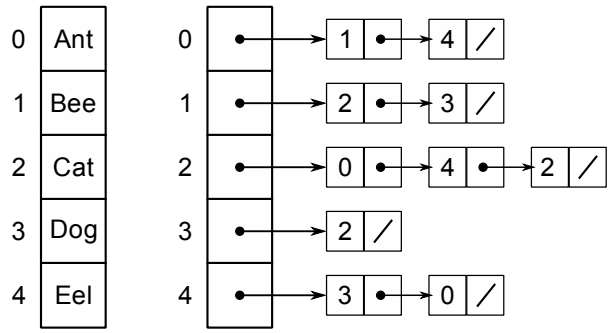
(Question 7 continued on next page)

(Question 7 continued)

```
    public boolean equals(Object obj){









    }
```

Question 8. Graphs [14 marks]

(a) [4 marks] Draw a diagram (circles and lines) of the directed graph corresponding to the following adjacency list representation of a graph. The array on the left contains the node labels.



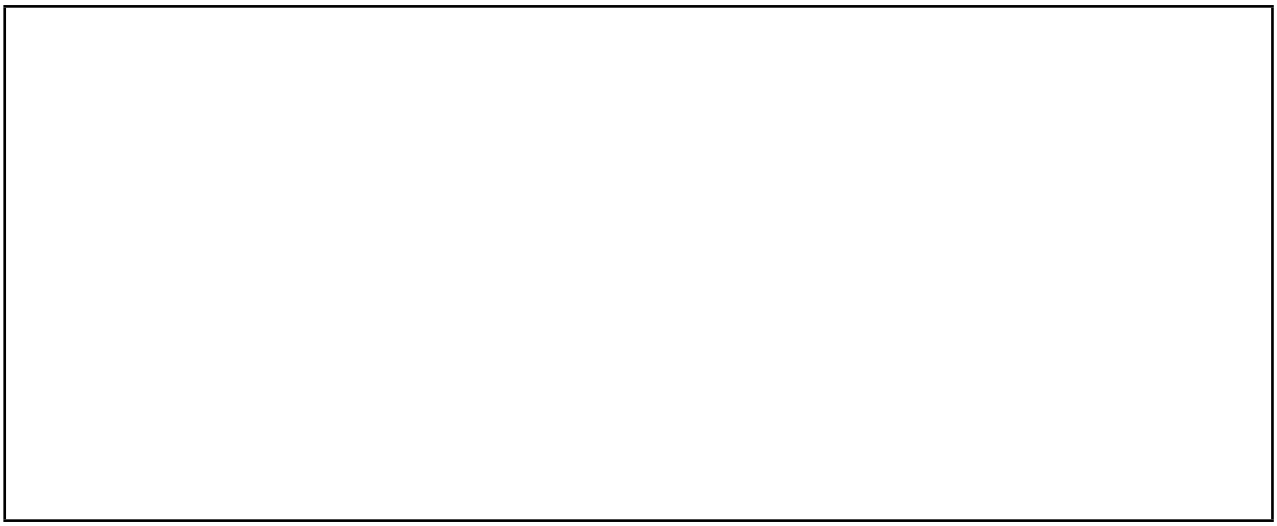
(Question 8 continued on next page)

(Question 8 continued)

(b) [4 marks] Draw a diagram (circles and lines) of the weighted undirected graph corresponding to the following adjacency matrix representation of a graph. The array on the left contains the node labels.

0	Ant
1	Bee
2	Cat
3	Dog
4	Eel

	0	1	2	3	4
0		18		43	
1	18		38	21	
2		38			
3	43	21			47
4				47	



(Question 8 continued on next page)

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

(Question 8 continued)

(c) [6 marks] Complete the following reachable method that will return true if there is a path from one node to another node in a directed, unweighted graph represented by an adjacency list, and will return false if there is no such path. Assume the following declarations for two fields containing the graph:

```
private String[] nodes;
private List<Integer>[] edges;
```

Assume that the arrays have been constructed and filled.

```
public boolean reachable(int node1, int node2){




}
```

****

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

(You may detach this page)

Appendices

Possibly useful formulas:

- $1+2+3+4+\cdots+k \;=\; \frac{k(k+1)}{2}$
- $1+2+4+8+\cdots+2^k \;=\; 2^{k+1}-1$
- $a+(a+b)+(a+2b)+\cdots+(a+kb) \;=\; \frac{(2a+kb)(k+1)}{2}$
- $a+as+as^2+as^3+\cdots+as^k \;=\; \frac{as^{k+1}-a}{s-1}$

Table of base 2 logarithms:

n	1	2	4	8	16	32	64	128	256	512	1024	1,048,576
$\log_2(n)$	0	1	2	3	4	5	6	7	8	9	10	20

Brief (and simplified) specifications of relevant interfaces and classes.

```
public class Random
   public int nextInt( int n);              // return a random integer between 0 and n−1
   public double nextDouble();              // return a random double between 0.0 and 1.0

public interface Iterator<E>
   public boolean hasNext();
   public E next();
   public void remove();

public interface Iterable<E>                // Can use in the "for each" loop
   public Iterator<E> iterator();

public interface Comparable<E>              // Can compare this to another E
   public int compareTo(E o);

public interface Comparator<E>              // Can use this to compare two E's
   public int compare(E o1, E o2);

DrawingCanvas class:
   public void drawLine(int x, int y, int u, int v)     // Draws line from (x, y) to (u, v)
   public void drawOval(int x, int y, int wd, int ht)   // Draws outline of oval
   public void drawString(String str, int x, int y)     // Prints str at (x, y)
```

```
public interface Collection<E>
    public boolean isEmpty();
    public int size();
    public boolean contains(Object item);
    public boolean add(E item);                    // returns false if failed to add item
    public Iterator<E> iterator();

public interface List<E> extends Collection<E>
    // Implementations: ArrayList
    public E get(int index);
    public void set(int index, E element);
    public void add(int index, E element);
    public void remove(int index);
    public void remove(Object element);

public interface Set extends Collection<E>
    // Implementations: ArraySet, SortedArraySet, HashSet
    public boolean contains(Object element);
    public boolean add(E element);
    public boolean remove(Object element);

public interface Queue<E> extends Collection<E>
    // Implementations: ArrayQueue, LinkedList
    public E peek();                               // returns null if queue is empty
    public E poll();                               // returns null if queue is empty
    public boolean offer(E element);

public class Stack<E> implements Collection<E>
    public E peek();                               // returns null if stack is empty
    public E pop();                                // returns null if stack is empty
    public E push(E element);                      // returns element

public interface Map<K, V>
    // Implementations: HashMap, TreeMap, ArrayMap
    public V get(K key);                           // returns null if no such key
    public V put(K key, V value);                  // returns old value, or null
    public V remove(K key);                        // returns value removed, or null
    public boolean containsKey(K key);
    public Set<K> keySet();                        // returns set of all keys in Map
    public Collection<V> values();                 // returns collection of all values
    public Set<Map.Entry<K, V>> entrySet(); // returns set of (key-value) pairs

Scanner class:
    public boolean hasNext()                       // Returns true if there is more to read
    public boolean hasNextInt()                    // Returns true if the next token is an integer
    public String next()                           // Returns the next token (chars up to a space/line)
    public String nextLine()                       // Returns string of chars up to next newline
    public int nextInt()                           // Returns the integer value of the next token
```