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

Student ID: .....

EXAMINATIONS — 2011

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 one 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 some relevant Java classes, interfaces, and exceptions can be found at the end of the paper.

Questions	**Marks**
1. Basic Questions	[27]
2. Using Collections	[15]
3. Implementing Collections	[15]
4. Recursion, and Sorting	[16]
5. Linked Lists, and Trees	[36]
6. Binary Search Trees	[24]
7. Partially Ordered Trees and Heaps	[26]
8. Bitsets and Hashing	[21]

Question 1. Basic questions [27 marks]

(a) [2 marks] List two advantages of using ArrayList over Arrays.

(b) [2 marks] Of the slower sorting sorting algorithms, which one has the best Big-O cost on almost sorted arrays? What is this cost?

(c) [2 marks] What is the depth of a binary tree whose maximum number of leaves is 16?

(d) [2 marks] Why is the contains() operation significantly faster to perform on a SortedArraySet than on a plain (unsorted) ArraySet?

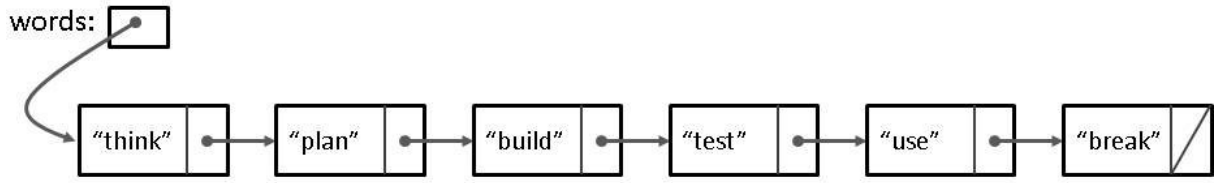
(e) [10 marks] For each example on the left, draw a line to the Collection type on the right that would be best for representing it.

The document files in an inbox tray.	SET
The keys on a keychain.	QUEUE
Things to do, written in a dairy.	STACK
Passengers boarding a plane.	MAP
Cars and their registration numbers.	LIST

(f) [2 marks] Which implementation of a Set discussed in this course has the best average performance for add, remove, and find?

(g) [3 marks] Write down the 3 main steps involved in the ensureCapacity() method of an ArrayList (which enables ArrayList to be "infinitely" stretchable).

Consider the diagram below of a variable words containing the reference of a linked list. Assume that each node of the list has the fields value and next.



(h) [2 marks] What is the value of words.next.value?

(i) [2 marks] What is the value of words.next.next.next.value?

Question 2. Using Collections [15 marks]

```
public class Customer{

    private String name;
    private Integer quantity;
    private String size;
    private String topping;
    :
    :
    :
    public Customer(String nName, Integer qQuantity, String sSize, String tTopping){
            name = nName;
            quantity = qQuantity;
            size = sSize;
            topping = tTopping;
    }

    public String toString(){
        return name+" has ordered "+quantity+" "+size+" "+topping+" pizza(s)";
    }

}
```

AngelsPizza is a popular Pizza joint in town. They need a simple system to simulate their customer orders. The program consists of a class AngelsPizza, which begins by initialising an ArrayQueue of Customer objects as follows:

```
public class AngelsPizza{
    private ArrayQueue<Customer> customers = new ArrayQueue<Customer>();
```

The class needs to have 2 methods, for loading customer details and for printing the details of the next customer in line to be served.

(a) [10 marks] Complete the loadCustomers method, which takes a String as the name of the file containing customer details, reads from that file, constructs new Customer objects (see Customer class on facing page), and adds them to the ArrayQueue. The file with the customer details has the following format:

```
Bob Smith 1 L Margherita
Jane Austen 2 S Chicago Style
P.G. Woodehouse 4 XL Seafood
Michael Clark 1 M Fruit
Susan Timon 2 L BBQ Chicken
```

```
public void loadCustomers(String fname){
    try{
        Scanner f = new Scanner(new File(fname));
        //YOUR CODE HERE









        }catch(IOException ex) { System.err.println(ex.getMessage());}
}
```

(b) [5 marks] Complete the serveCustomer method, which prints out the details of the next Customer from the ArrayQueue, or prints that there are no customers to serve, if the ArrayQueue is empty.
Note: the Customer class is written for you on page 4 and defines a toString() method that returns the customer details to be printed.

Sample printouts:

```
Next customer : Bob Smith has ordered 1 L  Margherita pizza(s)
There are no customers to serve.
```

```
public void serveCustomer(){
    //YOUR CODE HERE









}
```

Question 3. Implementing Collections [15 marks]

A local Rugby club maintains the Rugby World Cup (RWC) rankings for all top Rugby teams. They would like a simple system for maintaining rankings of the teams. The program consists of a class RWC, which begins by initialising a Map as follows:

```
public class RWC{
    private Map <Team, Integer> ratings = new HashMap<Team, Integer>();
```

The class needs to have 3 methods, for adding teams, removing teams, and printing all the teams along with their ratings.

(a) [5 marks] Complete the addTeam method, which adds the team along with their rating to the Map. It should either print out that the team has been added, or that the team already exists in the system. Assume that the Team class has been written for you and already defines a toString method that returns the name of the team (example: All Blacks, or Wallabies).

Sample printouts:

```
Team All Blacks have been added
Team Wallabies have been added
All Blacks already exist in the system!
```

```
public void addTeam(Team myTeam, Integer rating){
//YOUR CODE HERE









}
```

(b) [5 marks] Complete the removeTeam method, which removes the given team and prints out that it has been removed, or prints out that the team is not in the system if that team does not exist. Sample printouts:

```
Springboks have been removed!
Powerpuff Boys are not in the system.
```

```
public void removeTeam(Team myTeam){
//YOUR CODE HERE









}
```

(c) [5 marks] Complete the printAllTeams method, which prints out all the teams and their ratings. Assume the Team class is written for you and already defines a toString method that returns the name of the team (example: All Blacks or Wallabies). Sample printout:

```
RWC ratings
All Blacks: 1
Wallabies: 2
Brave Hearts: 6
Springboks: 5
----------------------
```

```
public void printAllTeams(){

        //  YOUR CODE HERE









}
```

Question 4. Recursion and Sorting [16 marks]

(a) [10 marks] A pebble is drawn as a black oval with a white outline, whose the height is $\frac{1}{4}$ its width. Write a **recursive** drawPebbles method that draws a sequence of pebbles, starting at (x,y), with the first pebble of the given width, and each successive pebble 80% the width of the previous one. The method should not draw any pebbles smaller than 15 in width.

[Note: remember to give a *recursive* solution]

```
public void drawPebbles(double x, double y, double width) {
        // YOUR CODE HERE




}
```

(b) [3 marks] Given the initial array shown (1st row), write down the array as it would be after **3 passes** through the outer loop of Selection Sort (*assume the usual A-to-Z sort is required*).

Start:	S	B	R	O	Q	D	F

After 3 passes of Selection sort:

(c) [3 marks] Given the initial array shown below (1st row), write down the array as it would be after **3 passes** through the outer loop of Insertion Sort (*assume the usual A-to-Z sort is required*).

Start:	M	P	L	Z	A	D	Q

After 3 passes of Insertion sort:

Question 5. Linked Lists and Trees [36 marks]

Consider the following declaration of LinkedNode, which can be thought of as the first node in a linked list.

```
public class LinkedNode<E> {
  // fields and constructor
  private E value;
  private LinkedNode<E> next;
  public LinkedNode(E val, LinkedNode<E> nd) {
     value = val;
     next = nd;
  }
  public void setValue(E item) { value = item;}      // the setters and getters
  public void setNext(LinkedNode<E> nd) { next = nd;}
  public E getValue() {return value;}
  public LinkedNode<E> getNext() { return next;}
}
```

(a) [4 marks] Complete the size method for the above LinkedNode class, that returns the number of items in the linked list which starts at the current node. You must give a *recursive* version.

```
public int size() {



}
```

(b) [8 marks] Suppose you are writing code to generate a chain of LinkedNodes, each containing an integer. The first node should contain the integer 5000, and each successive node should contain a value that is half (via integer division) the value stored in the preceding node. The chain should end when the value reaches 1 (*ie.* the final node will contain "1"). Write code to accomplish this.

```
   int n = 5000;









.
```

(c) [1 mark] What value would be returned by calling the size() method on the first LinkedNode from the previous question?

(d) [2 marks] What is the main problem with using LinkedNode on its own, as an implementation of the *List* interface?

(e) [5 marks] In lectures you met the idea of LinkedList as a way to implement the *List* interface. A LinkedList uses objects of the LinkedNode class to store the items.

Consider instead using a class which we shall call TreeList to implement the *List* interface by storing the items in a binary tree instead. **Discuss** any advantages (and / or disadvantages) you can see for this idea.

(f) [2 marks] A full binary tree with a depth of 10 has $2^{10} = 1024$ leaves on its bottom layer. How many *other* nodes are there in the tree?

(g) [2 marks] The term "Ternary" refers to 3 in the way that "Binary" refers to 2. What is the depth of a complete, balanced *ternary* tree having 41 nodes?

(h) [8 marks] Consider the following TernaryTreeNode class:

```
public class TernaryTreeNode <E> {
    private String word;
    private TernaryTreeNode <E> firstChild;
    private TernaryTreeNode <E> secondChild;
    private TernaryTreeNode <E> thirdChild;
      :
      :
}
```

In the box below, complete the find method in a TernaryTreeNode class. This should use recursion to search the tree below node for a node having a field word matching the argument text. It should return a reference to this node.

```
private TernaryTreeNode <E> find(TernaryTreeNode node, String text) {









}
```

The following is pseudocode for a particular tree traversal:

```
make a queue,
put root on the queue,
while queue not empty:
    poll a node off the queue
    process that node
    put its children on the queue
```

(i) [1 mark] Is this a recursive, or iterative, way of carrying out a traversal?

(j) [3 marks] What type of traversal would be performed by the above algorithm?

Question 6. Binary Search Trees [24 marks]

(a) [4 marks] Draw the *Binary Search Tree* that results from adding the following in the order given:

F, I, D, J, K, C, A, E, B

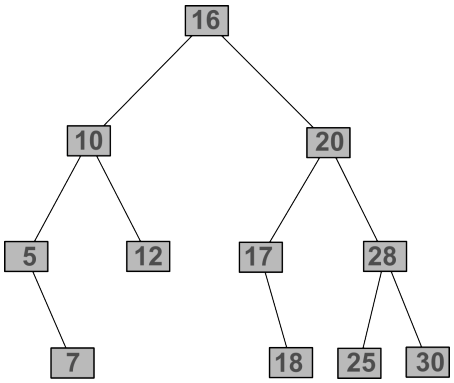
(b) [2 marks] What is special about an in-order traversal of a Binary Search Tree (BST)?

(c) [8 marks] As pseudocode, give an algorithm for conducting an **in-order** traversal of a binary tree, by making use of a Stack.

(d) [2 marks] What are "rotations", in the context of Binary Search Trees?

(e) [4 marks] Suppose that items consisting of the 7 letters `A` to `G` are to be added to a (BST) that is initially empty. Give an ordering of the letters such that, if the items are added in this order, the resulting BST will be perfectly balanced. (*NB.* You don't need to show the actual tree).

(f) [4 marks] In the box, draw the tree that results if the values 30, 5, 10, and 20 are removed (in that order) from the following BST.
Note there is spare working paper included with this exam.



Question 7. Partially Ordered Trees and Heaps [26 marks]

(a) [6 marks] Two efficient implementations of a priority queue are:

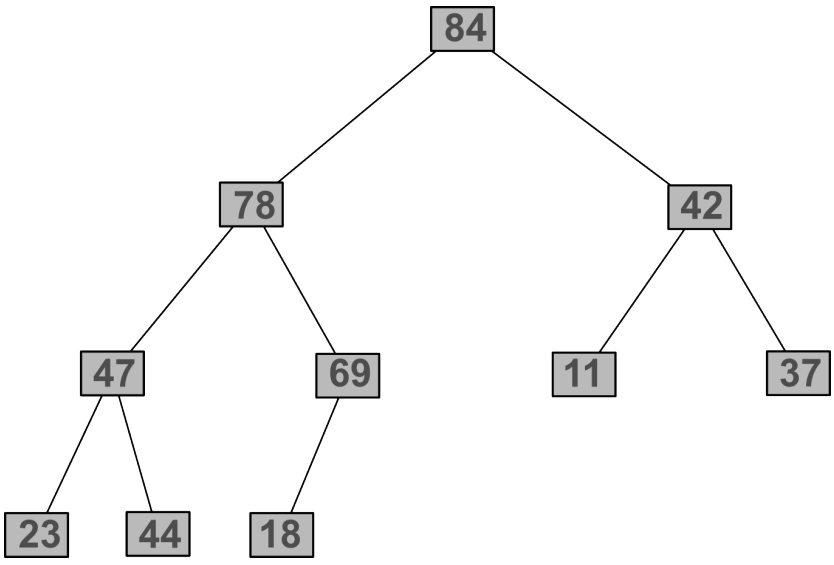
- an array of Queues, indexed by the priority, one queue for each priority value
- a HeapQueue, using a partially ordered tree in an array

Give one *advantage* and one *disadvantage* of using the first, compared to the second, implementation.

```
Advantage of using an array of Queues:




Disadvantage:
```



(b) [4 marks] Draw the *heap* (as an array) that corresponds to the above *POT*.

(c) [6 marks] Show the *POT* that results when the following values are added, in order, to the same tree (*ie.* to the original tree, not your answer to a previous question):

81, 25, 22, 86

(d) [4 marks] *As an array*, show the Heap that results from the following values being added, in order, to a *NEW* heap. Assume it is a "max" heap, with the largest value at the root.

3, 9, 11, 7, 5

(e) [6 marks] In words or pseudocode, give the algorithm for carrying out poll() on a Priority Queue that is implemented as a *Heap*.

Question 8. Bitsets and Hashing [21 marks]

(a) [2 marks] What does the term "load factor" mean, in the context of hashing?

(b) [4 marks] Lectures covered two different implementations for Hash Tables, namely "Chaining" (or buckets), and "Open Addressing". Deletions need to be handled differently in the two approaches. In words, describe how each implementation carries out deletions.

```
Chained implementation:



Open Addresssing implementation:
```

(c) [5 marks] Draw the Hash Table that results from adding the following hash codes to a hash table of size 7, using the division method and linked chaining: **11, 7, 4, 22, 74, 6**

(d) [5 marks] Draw the Hash Table from the previous question using a table size of 7, and open addressing using *linear probing*.

(e) [5 marks] Draw the Hash Table from the previous questions, using a table size of 7, and open addressing using *quadratic probing*.

****

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.

Appendix (may be removed)

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

```
public class UI extends java.lang.Object
    public static void clearGraphics();
    public static void setColor(java.awt.Color col);
    public static void drawOval(double x, double y, double width, double height);
    public static void fillOval (double x, double y, double width, double height);

public class Scanner
    public boolean hasNext(); // there is more to read
    public String next(); // return the next token (word)
    public String nextLine(); // return the next line
    public int nextInt(); // return the next integer

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);
```

```
public interface Collection<E>
    public boolean isEmpty();
    public int size ();
    public boolean add();
    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(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);

public interface Map<K, V>
    // Implementations: HashMap, TreeMap, ArrayMap
    public V get(K key);                 // returns null if no such key
    public void put(K key, V value);
    public void remove(K key);
    public Set<Map.Entry<K, V>> entrySet();
```