Name: . . . . . .

ID Number: . . . . . .

COMP103: Test

11 Aug, 2005.

Instructions

- Time: **2 hours**.
- Answer **all** the questions.
- There are 120 marks in total.
- Write your answers in the boxes in this test paper and hand in all sheets.
- Every box with a heavy outline requires an answer.
- If you do not understand a question, ask for clarification.
- There are useful formulas and documentation at the end of the exam paper.

		Marks		
1.	Collection Types	[16]	1	
2.	Using Collection Types and Classes	[12]	2	
3.	"Big O" Cost of Algorithms	[22]	3	
4.	Programming with Stacks	[11]	4	
5.	Programming with Maps	[10]	5	
6.	Implementing a Collection	[23]	6	
7.	Analysing Algorithms	[10]	7	
8.	Programming with LinkedNodes	[16]	8	
			Total:	

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 1. Collection Types [16 marks]

Briefly explain each type of collection below. For each type, specify whether the type imposes any structure on the values, whether it allows duplicates, and whether it has any constraints on how values can be accessed (or added, or removed).

Set:

Structure: No structure required

Duplicates: No duplicates allowed

Constraints: No constraints

List:

Structure: Items stored in a sequence/order

Duplicates: allowed

Constraints: No constraints

Stack:

Structure: Items stored in a sequence/order

Duplicates: allowed

Constraints: Only add and remove from one end

Queue:

Structure: Items stored in a sequence/order

Duplicates: allowed

Constraints: Add at one end, remove from the other end

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 2. Using Collection Types and Classes [12 marks]

Suppose you are writing a program dealing with medical centres in a region.

The main class has several fields that contain collections of information.

The program also contains a MedCentre class for storing information about an individual medical centre.

For each field described below, specify the type of the field and how the (empty) collection should be constructed. You can use any of the collection implementations that are in the Java Collections Framework or were described in the lectures or assignments. (See the appendix on the last page of the test.)

(a) [4 marks] The contacts field should contain a sequence of phone numbers to call in an emergency. Each phone number is represented by a String of digits and spaces.

```
private ..............................  contacts = new ..............................;
```

(b) [4 marks] The allCentres field should contain a collection of information about all the medical centres in a district. The program will look up a MedCentre in the collection using the name of the medical centre (a String) as a key.

```
private ..............................  allCentres = new ..............................;
```

(c) [4 marks] The deliveries field should contain a collection of delivery requests. Each delivery request will be an unordered collection of MedCentres to which the delivery must be made. When a delivery request is received, it will be added to the end of the deliveries collection. The requests are processed strictly in the order they were received.

```
private ..............................  deliveries = new ..............................;
```

Question 3. "Big O" Cost of Algorithms [22 marks]

Suppose an array contains n items. State the **worst** case asymptotic ("Big O") cost of each of the following:

(a) [2 marks] Searching for an item in the array using linear search.

Worst case cost = O(n)

(b) [2 marks] Searching for an item in the array using binary search (assuming the array is ordered).

Worst case cost = O($n \log(n)$)

(c) [2 marks] Sorting the array using Insertion Sort.

Worst case cost = O(n^2)

(d) [2 marks] Sorting the array using Merge Sort.

Worst case cost = O($n \log(n)$)

(e) [2 marks] Sorting the array using Quick Sort.

Worst case cost = O(n^2)

(Question 3 continued on next page)

(Question 3 continued)

(f) [4 marks] Suppose a program uses an algorithm with an average case asymptotic cost of $O(n)$, where n is the size of the input collection.

When you measure the program on a collection with 1,000 items, the running time is 1.2 seconds.

Give a reasonable estimate of the running time of the program on a collection with 100,000 items.

Justify your answer.

Estimated running time: 120 seconds = 2 mins

Justification: $O(n)$ means when the size increases by a factor of 200, the time should increase by a factor of 200.

(g) [4 marks] Suppose a program uses an algorithm with an average case asymptotic cost of $O(n^2)$, where n is the size of the input collection.

When you measure the program on a collection of 1,000 items, the measured running time is 15 milliseconds.

Give a reasonable estimate of the running time of the program on a collection of 100,000 items.

Justify your answer.

Estimated running time: 10000* 15 milli = 150 seconds = 2.5 minutes.

Justification: $O(n^2)$ means that if size increases by factor of 100 the time should increase by a factor of $100 \times 100 = 10,000$. Therefore the time should be 15*100*100 milliseconds.

(h) [4 marks] (Harder) On average, what fraction of the time will we require the worst case number of comparisons to find an item in a large array using binary search? Justify your answer.

Fraction: About 50% of the time.

Justification: Half the items in the array are at the bottom of the "tree" of comparisons that binary search follows.

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 4. Programming with Stacks [11 marks]

(a) [8 marks] Complete the following reverseFile method that reads a file, one line at a time, and then writes it out in reverse order. It should put each line on the Stack as it reads it. When it has finished reading, it should then remove each line from the Stack and print it out.

```
public void reverseFile(){
    Stack <String> lines = new Stack <String> ();
    String fname = FileDialog.open();
    try{
        Scanner s = new Scanner(new File(fname));









        s.close();









    }
    catch(Exception e){}
}
```

(b) [3 marks] Explain briefly why using a stack means that the file is printed in reverse order.

The stack is first in, first out, and last in, last out, so the last line to be pushed on the stack is the first line to be popped and printed.

Question 5. Programming with Maps [10 marks]

Suppose you are writing a Translator program to make a literal translations from English text to French. The field and constructor of the program are shown below.

The program first constructs a dictionary, and then uses the dictionary to translate a file of English words into French.

Complete the readDictionary and translate methods on the facing page.

```
import java.util.*;
import comp103.*;

public class Translator{

    Map <String, String> dictionary;    // Map of English words (keys) and French words (values)

    public Translator(){
        readDictionary();
        translate();
    }

        ⋮
```

Example dictionary file:

```
the le
a un
is est
small petit
chair chaise
pen plume
 :
```

Example input and output:

```
the green pen is on the chair

le green plume est on le chaise
```

(Question 5 continued on next page)

(Question 5 continued)

(a) [5 marks] The readDictionary method should read a file of English and French words into the dictionary field. Each line of the file contains one English word and the equivalent French word (see example on opposite page).

```
public void readDictionary(){
    dictionary = new HashMap <String, String> ();
    try{
        Scanner s = new Scanner(new File("EngFrench.txt"));
        while (s.hasNext()){









        }
        s.close();
    }
    catch(IOException e){}
}
```

(b) [5 marks] The translate method should read a file of English words, one word at a time. It should look up each English word in the dictionary, and if there is an associated French word, it should print the French word; otherwise it should print the English word.

```
public void translate(){
    try{
        Scanner s = new Scanner(new File(FileDialog.open()));
        while (s.hasNext()){









        }
        s.close();
    }
    catch(IOException e){}
}
```

Question 6. Implementing a Collection [23 marks]

A Bag is a type of collection that (like a Set) has no structure. Unlike a Set, a Bag is allowed to contain duplicates. Part of the code for the ArrayBag class is shown on the facing page.

(a) [3 marks] If an ArrayBag contains *n* items, what will be the average cost of the contains method?

O(*n*)

(b) [3 marks] If an ArrayBag contains *n* items, what will be the average cost of the add method?

O(1)

(c) [3 marks] Explain why this is different from the the cost of adding an item to an ArraySet.

Because a Bag can contain duplicates, we do not need to check whether the item is already present in the Bag.

(d) [6 marks] Complete the remove(Object item) below that will remove one element equal to item from an ArrayBag, and return true. If the ArrayBag does not contain any values equal to item, then remove will return false. Remember that it does not need to keep items in order.

```
public boolean remove(Object item){
    if (item == null) return false;









}
```

(Question 6 continued)

```
import java.util.*;
public class ArrayBag < E > extends AbstractCollection < E > {
    private static int INITIALCAPACITY = 10;
    private int count = 0;
    private E[ ] data;

    public ArrayBag(){
        data = (E[ ]) new Object[INITIALCAPACITY];
    }

    public boolean add(E item){
        if (item == null) return false;
        ensureCapacity();
        data[count] = item;
        count++;
        return true;
    }

    public boolean contains(Object item){
        for (int i = 0; i < count; i++){
            if (data[i].equals(item))
                return true;
        }
        return false;
    }

    public boolean remove(Object item){
        ⋮
    }

    private void ensureCapacity () {
        if (count < data.length) return;
        E[ ] newData = (E[ ]) new Object[data.length*2];
        for (int i = 0; i < count; i++)
            newData[i] = data[i];
        data = newData;
    }
```

(Question 6 continued on next page)

(Question 6 continued)

(e) [8 marks] (Harder) Complete the following ArrayBagIterator class that defines an iterator for an ArrayBag. Note that ArrayBagIterator is a private inner class of ArrayBag.

```
private class ArrayBagIterator < E > implements Iterator < E > {
    // Fields




    /** Constructor: argument is the ArrayBag to iterate down */
    private ArrayBagIterator (ArrayBag < E > b) {


    }
    /** Return true if iterator has at least one more element */
    public boolean hasNext () {


    }
    /** Return next element in the Bag */
    public E next () {









    }
    ** Remove from the set the last element returned by the iterator.
    * Can only be called once per call to next()./*
    public void remove(){









    }
}
```

Question 7. Analysing Algorithms [10 marks]

In the following code fragments, assume that data is a List of Strings and contains n values. Express your answers in terms of n.

(a) [3 marks] How many times will the following code fragment call the equals() method? (Do not use "Big-O" notation.)

```
int n = data.size();
for (int i = 0; i < n; i++){
    for (int j = 0; j < i; j++){
        if ( data.get(i).equals(data.get(j)) )
            data.set(i, "duplicate");
    }
}
```

$n(n-1)/2$

(b) [2 marks] Why would the data.set(...) step **not** be a good step to count in this code fragment?

It is in the inner loop, but it is inside an **if** and is not called every time.

(c) [5 marks] What is the asymptotic (Big-O) complexity of the following code fragment? Briefly justify your answer.

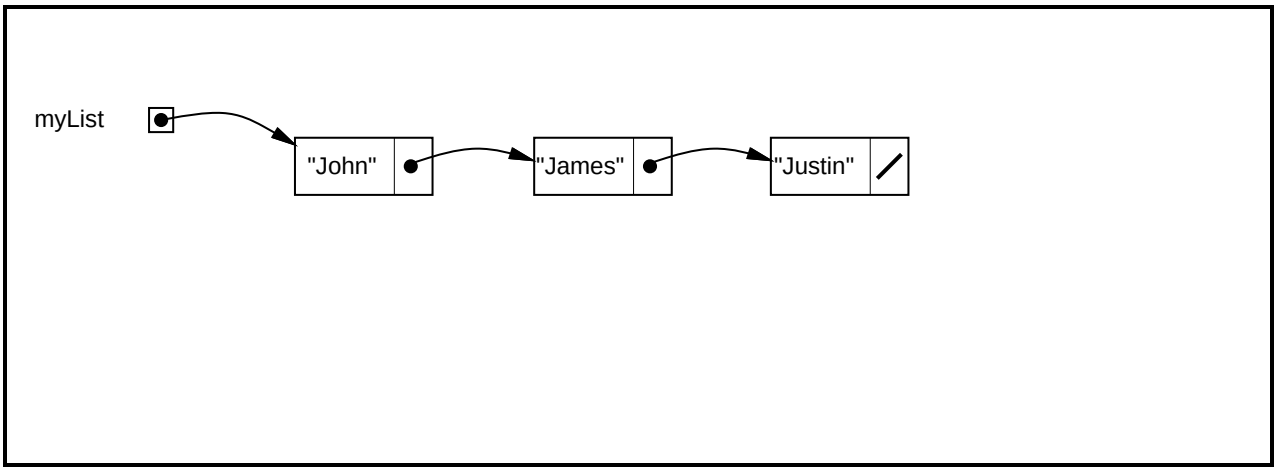
```
int n = data.size();
for (int i = n–1; i > 0; i--){
    for (int j = i; j > 0; j = j/2){
        System.out.println(data.get(i)+":"+data.get(j));
    }
}
```

Complexity: O($n \log(n)$)

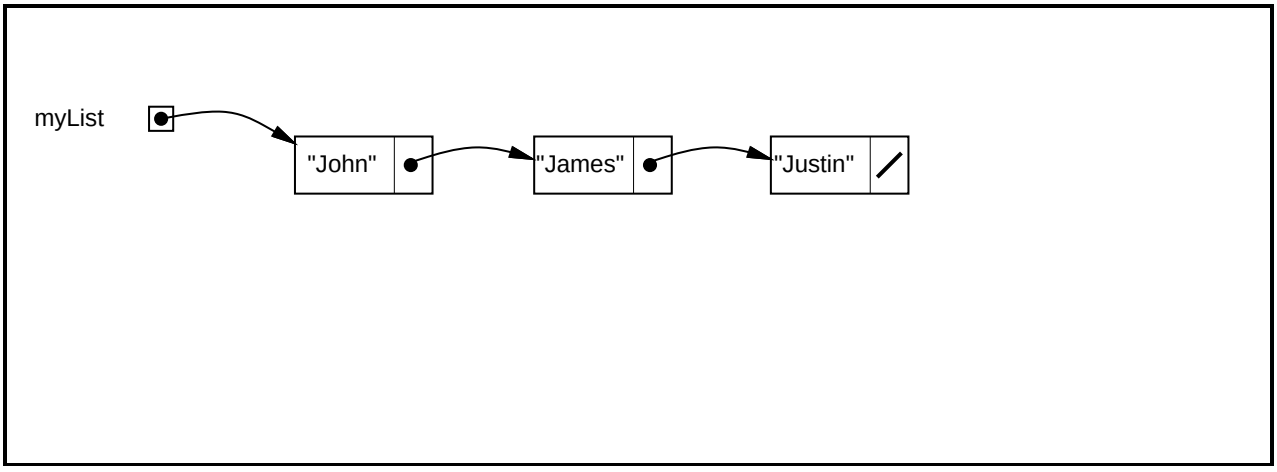
Justification: The inner loop will take $\log(i)$ steps. The outer loop repeats n times. so the cost is $\log(n-1) + \log(n-2) + \log(n-3) + \cdots + \log(1)$ which is at most $n \times \log(n)$ and at least $1/2n \log(n/2) = 1/2n \log(n) - 1/2n = \mathrm{O}(n \log(n))$

Question 8. Programming with LinkedNodes [16 marks]

(a) [3 marks] Consider the diagram below of a variable myList that contains a reference to a linked list of three items. On the diagram, draw the changes if the node containing `"John"` were removed from the list.



(b) [3 marks] Consider the same diagram of the same linked list. On the diagram, draw the changes if a new LinkedNode containing `"Jason"` were inserted into the list after the node containing `"James"`.



(Question 8 continued on next page)

(Question 8 continued)

Consider the following declarations for LinkedNode

```
public class LinkedNode<E> {
    private E value;
    private LinkedNode<E> next;
    public LinkedNode(Object v, LinkedNode<E> n){
        value = v;
        next = n;
    }
        ⋮
```

(c) [5 marks] Complete the following print method for the LinkedNode class that prints out each value in a linked list using System.out.println. You may give an iterative or recursive version.

```
public void print () {
    System.out.println(value);
        if (next != null)
            next.print();




    }
```

(d) [5 marks] (Harder) Complete the following reduce method for the LinkedNode class that removes every second node from a linked list. You may give an iterative or recursive version.

```
public void reduce () {
    if (next != null)
        next= next.next;
    if (next != null)
        next.reduce();




}
```

****

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
$\log(n)$	0	1	2	3	4	5	6	7	8	9	10	20

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

```
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 interface Iterator<E> {
    public boolean hasNext();
    public E next();
    public void remove();
}
public interface Comparator <E>{
    public int compare(E o1, E o2);
}
```

```
public interface Collection <E>{
    public boolean isEmpty ();
    public int size ();
    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 (E element);
    public void add (E element);
    public void remove (Object 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 ();
}
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 class Arrays {
    public static <E> void sort(E[ ] ar, Comparator<E> comp);
}
public class Collections {
    public static <E> void sort(List<E> list, Comparator<E> comp);
}
```

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.