

Family Name:

Other Names:

ID Number:

COMP103 Test

1st Sept, 2008

Instructions

- Time: **90 minutes**.
- Answer **all** the questions.
- There are 90 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	[17]	1	<input type="text"/>
2. Using Collections	[20]	2	<input type="text"/>
3. Implementing a Collection	[10]	3	<input type="text"/>
4. Sorting Algorithms	[19]	4	<input type="text"/>
5. Cost of Algorithms	[16]	5	<input type="text"/>
6. Linked Lists	[8]	6	<input type="text"/>
		Total:	<input type="text"/>

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

[17 marks]

(a) [12 marks]

Complete the following table of properties of the standard collection abstract data types.

Collection type	Ordered?	Duplicates?	Restricted access? (describe)
Bag	no	yes	no restrictions
Set			
Queue			
Stack			
List			

(b) [2 marks] Maps are more complex than just collections of items, in that they have both keys and values. Are duplicate keys, or duplicate values, allowed in a map?

Duplicate keys allowed? Duplicate values allowed?
--

(c) [3 marks] In Java, collections can only contain Objects. How does Java allow for collections whose elements are primitive types (int for example)?

--

Question 2. Using Collections

[20 marks]

Consider the following line of code that declares and initializes a variable containing a set of queues of shapes.

```
Set <Queue<Shape>> allShapes = new HashSet <Queue<Shape>> ();
```

(a) [2 marks] On the right hand side, why is `HashSet` used, rather than just `Set`?

(b) [2 marks] On the left hand side, why is `Set` used, rather than `HashSet`?

(Question 2 continued on next page)

(Question 2 continued)**(c)** [16 marks]

People learning a new language need to practice their comprehension, for example by reading a simple paragraph of text. It might be useful to have a list of the vocabulary used in the paragraph, grouped by word length.

Complete the following method that takes an argument paragraph that is a List of the words as they occur in the text. The method returns a List of Sets of words: the first item consists of the set of all words of length 1 in the paragraph, the second item is the set of all words of length 2, and so on. Assume that the longest word is at most 15 characters long.

```
public List<Set<String>> produceVocabSets (List<String> paragraph) {
```

```
}
```

Question 3. Implementing a Collection

[10 marks]

(a) [5 marks] The `ArraySet` class defines a `Set` collection that stores the items in an array. The header and fields of the class are given below:

```
public class ArraySet <E> extends AbstractSet <E> {  
    private static int INITIALCAPACITY = 10;  
    private int count = 0;  
    private E [] data = (E[]) (new Object[INITIALCAPACITY]);
```

The class uses a method to copy the items to a new larger data array if the current array is full. The best way to implement this method is to “double and copy”. A not so good way is to make a new array that is a fixed amount larger than the current one.

Complete the code for `ensureCapacityBad`, which uses this not-so-good way.

```
private void ensureCapacityBad () {
```

```
}
```

(b) [5 marks] (*Hard*) By calculating the amortised (average) “big-O” cost of incrementally building up an array of size n using the above method, show that `ensureCapacityBad` is much less efficient than the “double and copy” version.

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.

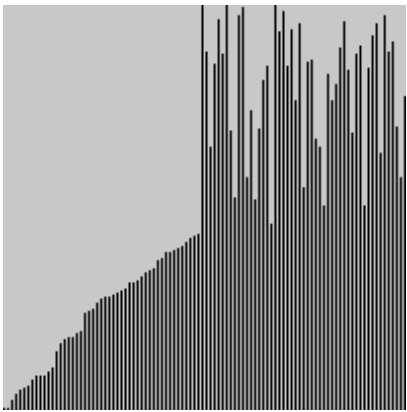
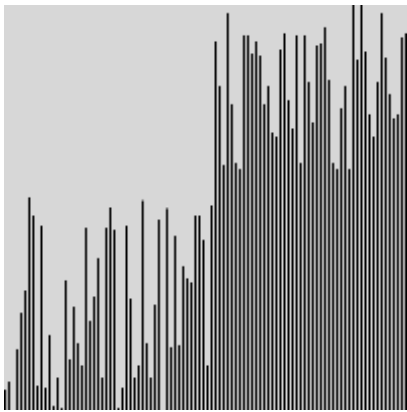
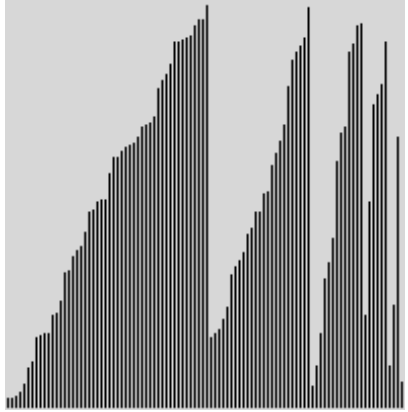
Question 4. Sorting Algorithms

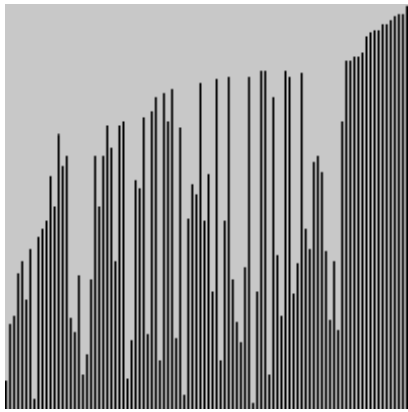
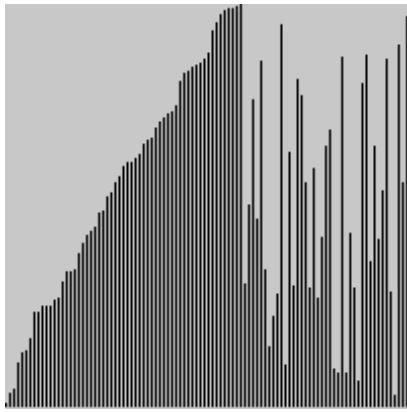
[19 marks]

The lectures gave code for five common sorting algorithms:

- BubbleSort
- MergeSort
- InsertionSort
- QuickSort
- SelectionSort

(a) [15 marks] Each of the examples below shows a different sorting algorithm caught part of the way through its operations on an array. Use your knowledge of the algorithms to decide which is being used in each case. Each algorithm appears once only. Write your answer below each picture.

 <p>Algorithm: Justification:</p>	 <p>Algorithm: Justification:</p>	 <p>Algorithm: Justification:</p>
---	--	---

 <p>Algorithm: Justification:</p>	 <p>Algorithm: Justification:</p>
--	---

(b) [2 marks] One algorithm performs much better when most of the items are already in order. Which one?

(c) [2 marks] Name ONE of the algorithms that is *stable* (i.e., keeps equal items in their original order).

Question 5. Cost of Algorithms

[16 marks]

(a) [2 marks] `ArraySet` uses an array to implement a collection that is not allowed to contain any duplicate items. We often need to check that a given item is not already in the collection, using the `contains` method. The average amount of time taken to do this check can be reduced if the items are stored in the array *in order*. What is the cost ("big-O" complexity) of checking `contains` on a set of n items, for the unsorted and sorted cases? (Note: ignore the cost of the sorting itself).

Complexity if unsorted: $O(\quad)$

Complexity if sorted: $O(\quad)$

(b) [2 marks] Explain how the sorted case can be done more efficiently.

The following `printInts` method prints out integers. When called with `printInts(8)` it produces the output shown on the right.

```

public void printInts ( int n){
    int k=0;
    for ( int i=1; i<=n; i=i+1) {
        for ( int j=0; j<i; j++) {
            k++;
            System.out.printf ("%3d ",k);
        }
        for ( int j=i; j<n; j++)
            System.out.printf (" * ");
        System.out.printf ("\n");
    }
}

```

1	*	*	*	*	*	*	*	*
2	3	*	*	*	*	*	*	*
4	5	6	*	*	*	*	*	*
7	8	9	10	*	*	*	*	*
11	12	13	14	15	*	*	*	*
16	17	18	19	20	21	*	*	*
22	23	24	25	26	27	28	*	*
29	30	31	32	33	34	35	36	*

(c) [4 marks] What is the asymptotic ("Big-O") complexity of `printInts`, as a function of n , and why?

Complexity: $O(\quad)$

Justification:

(Question 5 continued on next page)

(Question 5 continued)

Now consider the modified version of printInts(8) below. The only change is to the first for loop:

```

public void printInts ( int n){
    int k=0;
    for ( int i=1; i<=n; i=i*2) { // CHANGED
        for ( int j=0; j<i; j++) {
            k++;
            System.out.printf ("%3d ",k);
        }
        for ( int j=i; j<n; j++)
            System.out.printf (" * ");
        System.out.printf ("\n");
    }
}

```

1	*	*	*	*	*	*	*
2	3	*	*	*	*	*	*
4	5	6	7	*	*	*	*
8	9	10	11	12	13	14	15

(d) [4 marks] What is the "Big-O" complexity of this printInts, and why?

Complexity: This one is $O(\quad)$
 Justification:

The following version of printInts is the same, except that it doesn't print out any stars ("*").

```

public void printInts ( int n){
    int k=0;
    for ( int i=1; i<=n; i=i*2) {
        for ( int j=0; j<i; j++) {
            k++;
            System.out.printf ("%3d ",k);
        }
        System.out.printf ("\n");
    }
}

```

1							
2	3						
4	5	6	7				
8	9	10	11	12	13	14	15

(e) [4 marks] What is the "Big-O" complexity of this printInts, and why?

Complexity: $O(\quad)$
 Justification:

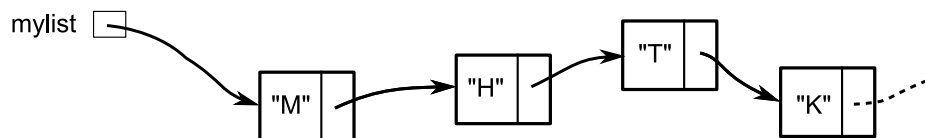
Question 6. Programming with LinkedNodes

[8 marks]

This question uses a `LinkedList` class, like the one in lectures, which defines `LinkedList` objects that contain a value and a reference to another `LinkedList`. The class is given on the facing page.

In your answers below, you should use the `getValue()`, `getNext()`, and `setNext()` methods, rather than the `value` and `next` fields.

Suppose `myList` is a variable of type `LinkedList<String>`, which contains a linked list of `LinkedList`s. For example, it might contain:



(a) [3 marks] Write a statement that will insert the string "A" at the second position of the list in `myList`. For the example above, `myList` should insert "A" between "M" and "H".

(b) [5 marks] Complete the following `lastValue` method that is passed a linked list whose values are string objects. It should return the value of the last item in the list. You may write a recursive or iterative version. The `LinkedList` class is given on the next page.

```

public String lastValue(LinkedList<String> list){
    if ( list == null) throw new RuntimeException("list is empty");

}
  
```

The `LinkedList` class:

```
public class LinkedList <E>{  
    private E value;  
    private LinkedList<E> next;  
  
    public LinkedList(E item, LinkedList<E> nextNode){  
        value = item;  
        next = nextNode;  
    }  
    public E getValue(){  
        return value;  
    }  
    public LinkedList<E> getNext(){  
        return next;  
    }  
    public void setNext(LinkedList<E> n){  
        next = n;  
    }  
}
```

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.

Appendices (may be removed)

Possibly useful formulas:

- $1 + 2 + 3 + 4 + \dots + k = \frac{k(k+1)}{2}$
- $1 + 2 + 4 + 8 + \dots + 2^k = 2^{k+1} - 1$

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 some relevant interfaces and classes.

```
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();
```