Family Name: ..... **Other Names:** .....

ID Number: ..... **Signature** .....

COMP 103: Mid-term Test

24th August, 2017

Instructions

- Time allowed: **45 minutes**
- There are 45 marks in total.
- Answer **all** the questions.
- Write your answers in the boxes in this test paper and hand in all sheets.
  You may tear off the last Java library documentation sheet, though.
- You may write notes on this paper, but make sure it is clear what and where your answers are.
- This test will determine 20% of your final grade
  (but your result will be boosted up to your exam result if the latter is higher).
- You may use paper translation dictionaries. No other aids are allowed.

Questions	**Marks**	
1. Various	[10]	
2. Programming with Collections	[10]	
3. Cost	[10]	
4. Linked Lists and Recursion	[15]	
TOTAL:	45	

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. Various questions [10 marks]

(a) [4 marks] By drawing a circle around the correct answer, indicate whether the following statements are TRUE or FALSE:

(i) AbstractList is a class, that extends List: TRUE / FALSE

(ii) The implements relationship is not allowed between interfaces: TRUE / FALSE

(iii) ArraySet and SortedArraySet have the same performance for add(item): TRUE / FALSE

(iv) Binary search depends on a list that is sorted and has no duplicates: TRUE / FALSE

(b) [2 marks] Suppose you use a variable playList of type List to keep a record of what songs you like to hear, with songs being objects of type Song. Write code to declare and initialise an empty play list.

(c) [4 marks] Name two similarities between stacks and queues (beyond that they are both collections) and state their respective principles.

Similarity 1:
Similarity 2:
Stack principle:
Queue principle:

Question 2. Programming with Collections [10 marks]

(a) [3 marks] What will the following code print?

```
public static void main(String[] a) {
    Stack<Integer> ss = new Stack<Integer>();
    ss.push(4);
    ss.push(3);
    UI.print(ss.peek());
    ss.push(2);
    ss.push(1);
    ss.pop();

    while (!ss.isEmpty())
        UI.print(ss.pop());
}
```

Output:

(b) [4 marks] For each of the lines below, either circle "YES" if the code is valid, or circle "NO" and explain the problem in the provided box.

(i) `Collection<LinkedNode> col1 = new ArrayList<LinkedNode> ();`

YES

NO

(ii) `Queue<Integer> col2 = new LinkedList<Integer> ();`

YES

NO

(iii) `LinkedNode<Collection> col3 = new LinkedNode<Collection> (null, null);`
(Question 4. shows a definition of LinkedNode, if you need a reminder.)

YES

NO

(iv) `List<List> col4 = new List<List> ();`

YES

NO

(c) [3 marks] Tom has written a class ReadingList so that he can represent book reading lists. A reading list can contain one or more books. Describe in words which steps Tom would have to undertake so that it becomes possible to treat a reading list like any other collection in the Java library, including being able to iterate through its books using Java's "for-each" syntax.

Question 3. Cost [10 marks]

(a) [2 marks] What is the average case complexity class of the following code, in terms of number of assignments?

```
for (int i=0; i<n; i++) {
    if (a[i] < 0) return;

    a[i] = b[i];
    b[i] = c[i];
}
```

O(_____)

(b) [2 marks] What is the average case complexity class of the following code, in terms of number of assignments?

```
for (int i=n−100; i>n−200; i−−) {
    a[i] = a[i−1];
}
```

O(_____)

(c) [2 marks] What is the average case complexity class of the following code, in terms of number of assignments?

```
for (int i=0; i<n/2; i++)
    for (int j=0; j<n/2; j++)
        a[i][j] = a[j][i];
```

O(_____)

(d) [2 marks] What is the average case complexity class of the following code, in terms of number of assignments?

```
int i=n;

while (i>0) {
    a[i] = a[ i / 2 ];
    i = i / 2;
}
```

O(_____)

(e) [2 marks] The web service "*electronic delights*" accepts order files containing details of orders. Each line of an order file contains a product name and its order quantity, e.g., "Huawei-P9-phone, 3". You have created a large order file with n lines, but a single line does not comply with the required format and the web service rejects the whole order as a result with the error code "*invalid line count: 1*". All n lines look fine to you but you need to figure out which line is the offending one. What is the **quickest** way of finding out which line caused the rejection? State the number of attempts you will need at most, and briefly describe the approach you would take.

Number of attempts needed at most:

Brief description of your approach:

Question 4. Linked Lists and Recursion [15 marks]

Assume a LinkedList class which uses the following LinkedNode implementation:

```
class LinkedNode<E> {

    // Data fields
    public E value;
    public LinkedNode<E> next;

    // Constructor
    public LinkedNode(E v, LinkedNode<E> n) {
        value = v;
        next = n;
    }

    E value() {return value;}
    LinkedNode<E> next() {return next;}
}
```

(a) [8 marks] Write a method **void** insertItemAfter(item, existingValue, list) (for class Linked**List**) which adds an item (item) to list, right after the position of the first node in list that contains existingValue. If existingValue does not occur in list then item should not be added.

You may assume that arguments item and existingValue are not null.

An ***iterative*** implementation of the method will give you at most five marks. Full marks are only awarded for a ***recursive*** method implementation.

```
void insertItemAfter(E item, E existingValue, LinkedNode<E> list) {









}
```

(b) [7 marks] Write a ***recursive*** method LinkedNode<*E*> remove(obsoleteValue) (for class Linked**Node**) which removes obsoleteValue from the receiver list. For example, myList = myList.remove(3); should result in myList having the same elements in the same order as before, but with the first occurrence of a 3 removed. If obsoleteValue does not occur in the list then the list should not be altered.

You may assume that obsoleteValue is not null.

```
LinkedNode<E> remove(E obsoleteValue) {




}
```

****

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

Some brief and truncated Java library documentation that may be helpful:

```
interface Collection<E>
    public boolean isEmpty()
    public int size()
    public boolean add(E item)
    public boolean contains(Object item)
    public boolean remove(Object element)
    public Iterator<E> iterator()

interface List<E> extends Collection<E>
    // Implementations: ArrayList, LinkedList
    public E get(int index)
    public E set(int index, E element)
    public void add(int index, E element)
    public E remove(int index)
    // plus methods inherited from Collection

interface Set extends Collection<E>
    // Implementations: ArraySet, HashSet, TreeSet
    // methods inherited from Collection

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)    // returns false if fails to add

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 being pushed

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 old value, or null
    public boolean containsKey(K key)
    public Set<K> keySet()

public class Collections
    public void sort(List<E>)
    public void sort(List<E>, Comparator<E>)
    public void shuffle(List<E>, Comparator<E>)
```