DUBLIN CITY UNIVERSITY

SEMESTER ONE EXAMINATIONS 2012

MODULE: *(Title and Code)*	CA213 Data Structures and Algorithms
COURSE:	B.Sc. in Computer Applications (CASE) B.Eng. in Digital Media Engineering B.Eng. in Information & Communications Engineering B.Eng. in Electronic Engineering
YEAR:	2/3
EXAMINERS:	Dr James Power Dr Frank Bannister Prof. J. M. Morris (Ext. 8419)
TIME ALLOWED:	2 Hours
INSTRUCTIONS:	***Answer any 4 questions.*** All questions carry equal marks. There are five questions in total. Answers may draw only on those elements of the Java library listed at the end of the paper. As instructed on answer books, candidates must not write using red ink.

Requirements for this paper
Please tick (X) as appropriate

- ☐ ***Log Table***
- ☐ ***Graph Paper***
- ☐ ***Attached Answer Sheet***
- ☐ ***Statistical Tables***
- ☐ ***Floppy Disk***
- ☐ ***Actuarial Tables***

THE USE OF PROGRAMMABLE OR TEXT STORING CALCULATORS IS EXPRESSLY FORBIDDEN

Please note that where a candidate answers more than the required number of questions, the ones to be marked should be clearly indicated; otherwise the first four will be marked.

PLEASE DO NOT TURN OVER THIS PAGE UNTIL ASKED TO DO SO

Question 1

(i) With respect to *interfaces* in Java, describe (a) the definition of an interface, and (b) its use in class definitions.

[6 marks]

(ii) Write an interface `Printable` which has a single method `put` whose intention is to print on the standard output a string representation of the object (i.e. the object whose class implements `Printable`).

[6 marks]

(iii) Enhance the given class `Date` below (encapsulating the notion of a *date*) so that it implements `Printable`. A date should appear on the standard output in a form typified by `15/3/2008`.

```
class Date {

  private int day,month,year; // day, month, year

  Date(){}

  void get(){ ... do not code ... }

  boolean lte(Date d) { ... do not code ... }

}
```

[6 marks]

(iv) Write a static method `print` which takes an array of objects whose class implements `Printable`, and prints each element in the array, one element per line.

[7 marks]

Question 2

(i) Explain briefly what is meant by a *map* in the context of data structures in Java.

[5 marks]

(ii) A *bag* is a collection of items in which each item can appear more than once. For example, a bag representing the forenames of persons in a room might contain 3 `Pat`s, 1 `John`, and 2 `Mary`s. A bag has operations to (a) add an item, (b) remove an item (if possible), and (b) determine the number of occurrences of an item. An outline implementation of a bag of `String`s is given below using Java's maps. Complete the implementation.

```
class Bag {  // a bag of strings

  private Map<String,Integer> map = ...; //the items
     // in the bag, and their number of occurrences

  void add(String w) {
  // add w to bag
     ...
  }

  int frequency(String w) {
  // number of occurrences of w in bag
     ...
  }

  boolean remove(String w) {
  // remove one occurrence of w if there is at least
  // one, and return true; otherwise return false.
     ...
  }
}
```

[20 marks]

Question 3

(i) Write an ADT for a bounded queue of elements of generic type `T`. It should have a no-args constructor, and the three basic queue methods, i.e. enqueue and dequeue an item, and determine if the queue is empty. Note that dequeuing is not possible if the queue is empty.

[10 marks]

(ii) Explain, with the help of a diagram, how a bounded queue is implemented using a "circular array" to store the items.

[10 marks]

(iii) Suppose you wish to convert the implementation you described in part(ii) to that of an *unbounded* queue. Which of the three methods in the implementation would you change, and how?

[5 marks]

Question 4

(i) A Java program uses a variable `t` to store a list of strings. Draw an illustration to indicate how the list is stored assuming the type of `t` is (a) `ArrayList<String>`, and (b) `LinkedList<String>` (lists are singly-linked in the case of (b)). For purposes of illustration, assume the list contains `pig cow cat dog hen` (in that order) .

[8 marks]

(ii) State the time complexity of `t.add(0,"rat")` in cases (a) and (b) above, and justify your answer (you may assume the list has `n` items).

[6 marks]

(iii) State the time complexity of `t.add("rat")` in cases (a) and (b) above, and justify your answer (again assume the list has `n` items).

[6 marks]

(iv) If you had need of a bounded queue in a program, which of (a) and (b) would you choose to use for the queue? Explain.

[5 marks]

Question 5

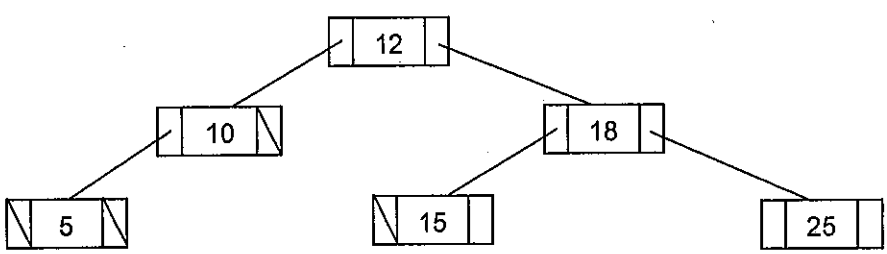
(i) The Java library provides a generic class `TreeSet<T>` to implement sets of elements of type `T`. What requirements, if any, are placed on `T`?

[6 marks]

(ii) What is a *binary tree*? What is a *binary search tree*?

[6 marks]

(iii) Draw the binary search tree after elements 16 and 21 are added to, and then 18 is removed from the binary search tree below (assuming the standard insertion and deletion algorithms).



[6 marks]

(iv) An outline implementation of class `TreeSet<T>` is provided below. It implements sets of elements of type `T` using binary search trees. Complete the code of `max` which returns the maximum value in the set, or `null` if there are no elements (e.g. if the set is as represented in part (iii) above, then 25 is returned).

```
class TreeSet<T extends Comparable<T>> {

   private static class Node<T> { // node of binary search tree
      private T item;    // data item
      private Node<T> left, right; //left and right subtrees

      Node(T item0, Node<T> left0, Node<T> right0)  {
         item = item0; left = left0; right = right0;
      }
   }
```

```
    private Node<T> root = null;      // root of tree
    private int numItems = 0;         // number of  nodes
    T max() { // maximum element (null if no elements)
       ...
    }
    // other methods omitted ...
}
```

[7 marks]

[End of exam]

Table of Java Classes *(Interfaces in italics)*

ArrayList<T>
ArrayList()
ArrayList(Collection<T>)
methods of List<T>

Boolean
Boolean(boolean)
boolean booleanValue()

Character
Character(char)
char charValue()
static boolean isDigit(char)
static boolean isLetter(char)
static boolean isLetterOrDigit(char)
static boolean isLowerCase(char)
static boolean isUpperCase(char)
static boolean isWhitespace(char)
static char toLowerCase(char)
static char toUpperCase(char)

Collection<T>
int size()
boolean add(T)
boolean remove(Object)
boolean contains(Object)
void clear()
boolean isEmpty()
boolean addAll(Collection<T>)
boolean retainAll(Collection<?>)
boolean removeAll(Collection<?>)
boolean containsAll(Collection<?>)
Iterator<T> iterator()

Collections
static void sort(List<T>)
static int binarySearch(List<T>, T)
static void shuffle(List<?>)

Comparable<T>
int compareTo(T)

Console
static int readInt()
static boolean readBoolean()
static double readDouble()
static char readChar()
static String readString()
static void skipLine()
static String readToken()
static boolean endOfFile()
static boolean hasMoreTokens()
static void skipWhitespace()

ConsoleReader
ConsoleReader(String)
int readInt()
boolean readBoolean()
double readDouble()
char readChar()
String readString()
void skipLine()
String readToken()
boolean endOfFile()
boolean hasMoreTokens()
void skipWhitespace()

DataInputStream
DataInputStream(FileInputStream)
int readInt() throws IOException
long readLong() throws IOException
boolean readBoolean() throws IOException
char readChar() throws IOException
double readDouble() throws IOException
float readFloat() throws IOException
String readUTF() throws IOException
int read(byte[]) throws IOException
int available() throws IOException
void close() throws IOException

DataOutputStream
DataOutputStream(FileOutputStream)
void writeBoolean(boolean) throws IOException
void writeInt(int) throws IOException
void writeLong(long) throws IOException
void writeDouble(double) throws IOException
void writeFloat(float) throws IOException
void writeChars(String) throws IOException
void writeUTF(String) throws IOException
void writeChar(int) throws IOException
void write(byte[], int, int) throws IOException
void close() throws IOException
void flush() throws IOException
int size()

Double
Double(double)
double doubleValue();
static double parseDouble(String)
static String toString(double)

Exception
void printStackTrace()

File
File(String)
boolean exists()
boolean isFile()
boolean isDirectory()
boolean canRead()
boolean canWrite()
long length()
boolean delete()
String getName()
boolean renameTo(File)

File[] listFiles()

FileInputStream

FileInputStream(String) throws
FileNotFoundException

FileOutputStream

FileOutputStream(String) throws IOException
FileOutputStream(String, boolean) throws
IOException

FileReader

FileReader(String) throws
FileNotFoundException

FileWriter

FileWriter(String) throws IOException
FileWriter(String, Boolean) throws
IOException

Float

Float(float)
float floatValue();
static float parseFloat(String)
static String toString(float)

GregorianCalendar

GregorianCalendar()
int get(int)
static int YEAR
static int MONTH
static int DATE
static int HOUR_OF_DAY
static int MINUTE
static int SECOND

HashMap<T,U>

HashMap()
HashMap(Map<T,U>)
methods in Map<T,U>

HashSet<T>

HashSet()
HashSet(Collection<T>)
methods in Set<T>

Integer

Integer(int)
int intValue()
static int parseInt(String)
static String toString(int)

Iterable<T>

Iterator<T> iterator()

Iterator<T>

boolean hasNext()
T next()
void remove()

LinkedList<T>

LinkedList()
LinkedList(Collection<T>)
void addFirst(T)
T getFirst()
T getLast()
T removeFirst()
methods in List<T>

List<T>

int size()
T set(int, T)
T get(int)
boolean add(T)
String toString()
void add(int, T)
T remove(int)
boolean remove(Object)
boolean contains(Object)
int indexOf(Object)
void clear()
boolean isEmpty()
boolean addAll(Collection<T>)
boolean retainAll(Collection<?>)
boolean removeAll(Collection<?>)
boolean containsAll(Collection<?>)
Iterator<T> iterator()

Long

Long(long)
long longValue()
static long parseLong(String)
static String toString(long)

Map<T,U>

void clear()
U put(T key, U value)
U get(Object key)
U remove(Object key)
boolean containsKey(Object key)
int size()
boolean isEmpty()
Set<T> keySet()
String toString()

Math

static double random()
static double abs(double)
static int abs(int) *et cetera*
static double ceil(double)
static double floor(double)
static int max(int, int) etc.
static int min(int, int) etc.
static double rint(double)
static long round(double)
static int round(float)
static double sqrt(double)

Object

boolean equals(Object)
String toString()

PrintWriter

PrintWriter(String)

PrintWriter(FileWriter)
PrintWriter(FileWriter, boolean)
void print(String)
void println(String)
void println()
PrintWriter printf(String, Object...)
void close()
void flush()

RandomAccessFile

RandomAccessFile(String, String) throws IOException
long length() throws IOException
void seek(long) throws IOException
int readInt() throws IOException
long readLong() throws IOException
boolean readBoolean() throws IOException
char readChar() throws IOException
double readDouble() throws IOException
float readFloat() throws IOException
String readUTF() throws IOException
void writeBoolean(boolean) throws IOException
void writeInt(int) throws IOException
void writeLong(long) throws IOException
void writeDouble(double) throws IOException
void writeFloat(float) throws IOException
void writeChars(String) throws IOException
void writeUTF(String) throws IOException
void writeChar(int) throws IOException
void close() throws IOException

Scanner

Scanner(File) throws FileNotFoundException
Scanner(String)
String nextLine()
String next()
int nextInt()
long nextLong()
double nextDouble()
boolean nextBoolean()
boolean hasNextLine()
boolean hasNext()
boolean hasNextInt()
boolean hasNextLong()
boolean hasNextDouble()
boolean hasNextBoolean()
void close()

Set<T>

boolean add(T)
boolean remove(T)
boolean contains(T)
int size()
void clear()
boolean isEmpty()
boolean addAll(Collection<T>)
boolean retainAll(Collection<?>)
boolean removeAll(Collection<?>)
boolean containsAll(Collection<?>)
Iterator<T> iterator()

String

int length()
boolean startsWith(String)
boolean startsWith(String)
boolean endsWith(String)
int indexOf(String)
String substring(int, int)
String substring(int)
char charAt(int)
String toUpperCase()
String toLowerCase()
String trim()
static String valueOf(int)
static String valueOf(long)
static String valueOf(double)
static String valueOf(float)
static String valueOf(boolean)
static String valueOf(char)
static String valueOf(char[], int, int)
static String valueOf(char[])
boolean equalsIgnoreCase(String)
void getChars(int, int, char[], int)

System

static long currentTimeMillis()
static long nanoTime()
static void exit(int)

TreeSet<T>

TreeSet()
TreeSet (Collection<T>)
methods in Set<T>

TreeMap<T,U>

TreeMap()
TreeMap(Map<T,U>)
methods in Map<T,U>