TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI

VICTORIA
UNIVERSITY OF WELLINGTON

Student ID: .......................

**EXAMINATIONS — 2010**

MID 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 | [17] |
| 2. Using Collections | [24] |
| 3. Implementing Collections | [20] |
| 4. Linked Structures | [18] |
| 5. Trees, and Binary Search Trees | [35] |
| 6. Partially Ordered Trees and Heaps | [29] |
| 7. Various topics: costs, sorting, hashing | [37] |

## 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. Basic Questions [17 marks]

**(a)** [2 marks] Which of the basic Collection types (Bag, Set, List, Stack, Queue, Map) would be best for representing the waiting line at an airline check-in desk?

**(b)** [3 marks] In a full binary tree with $N$ nodes, how many of those are leaf nodes?

**(c)** [2 marks] Why is it a bad idea to implement a Stack with a Linked List in which the top of the Stack is at the end of the List?

**(d)** [2 marks] A SortedArraySet is a better implementation to use than a plain (unsorted) ArraySet, because it speeds up the contains() operation. Why is this operation substantially faster in SortedArraySet?

**(e)** [2 marks] Why are add and remove operations *not* significantly faster in a SortedArraySet than in ArraySet?

**(f)** [6 marks] Consider the cost of contains in different implementations of Set. Give one implementation where the cost is $O(n)$, one that has a cost of $O(\log n)$, and one that has a cost of $O(1)$.

$O(n)$:

$O(logn)$:

$O(1)$:

## Question 2. Using collections [24 marks]

A supermarket system uses a Map from barcodes (stored as Integers) to shopping items. Shopping items are represented by the following very simple class:

```
public class ShopItem{
    private String description;
    private double cost;

    public ShopItem(String description, double cost) {
        this.description = description;
        this.cost = cost;
    }
    public String getDescription() { return this.description; }
    public void setDescription(String desc) { this.description = desc; }
    public double getCost() { return this.cost; }
    public void setCost(double c) { this.cost = c; }
}
```

**(a)** [12 marks] When a shopping trolley arrives at a check-out in the supermarket, items in the trolley have their barcodes read into a List.

Complete the code for a method CountCodes(), which takes a List of barcodes as input. Your method should return a Map from barcodes present in the List to counts of the occurrences of that barcode. Both of these are Integers. The map should contain one entry for each unique barcode that appears in the List.

```
    public Map <Integer,Integer> CountCodes (List<Integer> codes)
    {




    }
```

**(b)** [12 marks] Complete the PrintBill method below, which takes the Map returned by CountCodes *and* the supermarket's Map from barcodes to shopping items (Integer to ShopItem). The method generates a printout with one line for each type of item giving the item's description, the number of such items, and the per-item cost. This is followed by a final line giving the total cost. For example, if the supermarket's map is:

```
005   --->  Milk,      4.00
115   --->  Biscuits, 4.00
233   --->  Bread,     3.50
442   --->  Jaffas,    1.80
733   --->  Bag of Oranges,  8.50
```

and the map returned by CountCodes is

```
115   --->  1
233   --->  2
```

then PrintBill should print this:

```
Biscuits    1    $4.00
Bread       2    $3.50
TOTAL:          $11.00
```

```
public void PrintBill (Map<Integer,Integer> trolley, Map<Integer,ShopItem> codemap) {

}
```

## Question 3. Implementing Collections [20 marks]

**(a)** [4 marks] Each Collection type needs to be Iterable, which means their class must provide an Iterator. Give TWO reasons for making the items in a collection Iterable.

1:

2:

Consider the following code for a Pseudo Random Number generator:

```
public class RandNumIterator implements Iterator <Integer> {
  private int num=1;
  public boolean hasNext() {
    return true;
  }
  public Integer next() {
    num = (num ∗ 92863) % 104729 + 1;
    return num;
  }
  :
  :
}
```

**(b)** [2 marks] Why does this code include a method hasNext(), when all it does is return true?

**(c)** [6 marks] Complete the method below, that uses RandNumIterator to put 100 random integers into a List, which it returns.

```
public List <Integer> numberList () {









}
```

**(d)** [8 marks] ArrayList is an implementation of List that has two fields:

```
public class ArrayList <E> extends AbstractList <E> {
    private E[] data;
    private int count;
       :
}
```

Complete the code for the remove method of ArrayList, that removes the item at a specified index. The method should check that the index is valid, and should return the item that is removed.

```
public E remove (int index) {

}
```

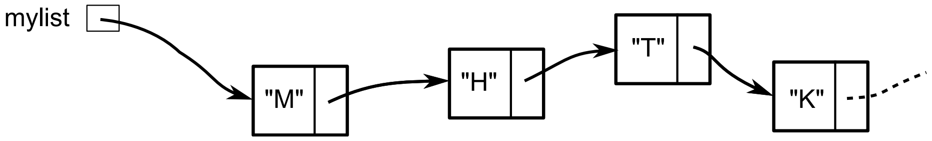
## Question 4. Linked Structures [18 marks]

This question uses a LinkedNode class, like the one in lectures, which defines objects that contain a value and a reference to another LinkedNode. The class is as follows:

```
public class LinkedNode <E>{
    private E value;
    private LinkedNode<E> next;

    public LinkedNode(E item, LinkedNode<E> nextNode){     // CONSTRUCTOR
        value = item;
        next = nextNode;
    }
    public E getValue(){ return value;   }
    public LinkedNode<E> getNext(){ return next; }
    public void setNext(LinkedNode<E> n){ next = n; }
}
```

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

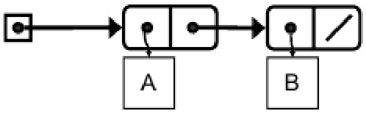
Suppose myList is a reference to the first LinkedNode<String> in a linked list. For example it might contain:

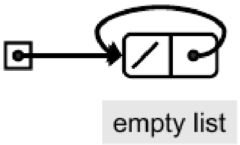


**(a)** [6 marks] By using the methods in LinkedNode, write a single statement that will insert the string "A" at the third position of the list refered to be myList. For the example above, mylist should insert "A" between "H" and "T". You may assume that the linked list has at least 4 items in it.

**(b)** [4 marks] Similarly, write a single statement that will remove the item at the second position of the list. For the example above, mylist should remove the item "H". Again, you may assume that the linked list has at least 4 items in it.

Linked lists composed of LinkedNodes could be used as a full List implementation, but to do so it is necessary to be able to call methods on a List that is empty. In lectures we mainly discussed the option of using a ”header” class for this reason. But we also considered denoting an empty list by a node with its next field set to the node itself, as indicated here:





To allow this, we first add an alternative constructor to the LinkedNode class above:

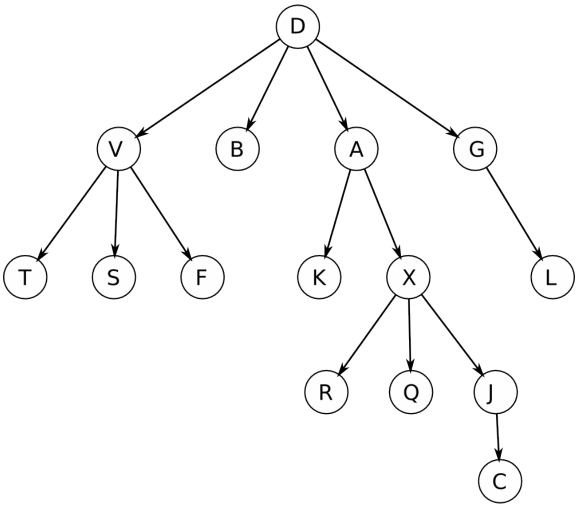
```
/* Alternative constructor: makes a new, empty, list */
public LinkedNode() {
    next = this;
}
```

**(c)** [8 marks] Complete the code for an addAtEnd() method for the LinkedNode class, which should use recursion to add a new item **to the end** of the list (of which the current LinkedNode is the start). Make sure your method copes with the case of an empty list, as described above.

```
public void addAtEnd(E item) {















}
```

**Question 5. Trees, and Binary Search Trees** [35 marks]

The next four questions concern the following tree:



**(a)** [2 marks] What is the depth of the tree shown above?

**(b)** [3 marks] List the order in which the nodes in the tree will be visited by a *breadth first* traversal.

**(c)** [3 marks] List the order in which the nodes in the tree will be visited by a *pre-order depth first* traversal.

**(d)** [3 marks] List the order in which the nodes in the tree will be visited by a pre-order, depth first traversal, *implemented iteratively with a stack, assuming that the children of a node are put on the stack in left-to-right order*.

**(e)** [5 marks] In the box below, show a tree consisting of 5 nodes A, B, C, D and E, for which a *pre-order* traversal yeilds the ordering ABCDE, but a *post-order* traversal gives BDECA.

**(f)** [5 marks] Draw the tree that would result from adding nodes to an empty Binary Search Tree in the following order:
G, V, F, S, Y, D, E, A, T, Q

**(g)** [1 mark] Is your tree (above) a *complete* tree?

**(h)** [5 marks] What is the ordering property that defines a Binary Search Tree?

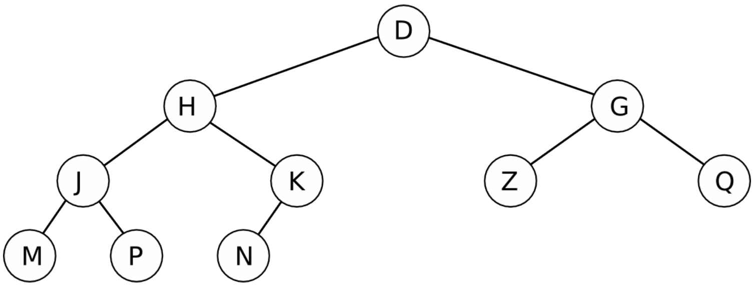
**(i)** [8 marks] Consider the remove operation on a Binary Search Tree. If the node being removed has only one child then it is fairly obvious what to do: simply "pulling up" that child results in a tree that is still a valid BST.

What should you do when removing a node that has two children? In your explanation, make it clear why this still results in a valid BST. *(Hint: You may find that drawing an example helps).*

Note: do not give java code for this question.

**Question 6. Partially Ordered Trees and Heaps** [29 marks]

Consider the following Partially Ordered Tree (POT):



**(a)** [3 marks] Draw the tree as a heap, in the row of boxes below.
(A Heap is a partially ordered binary tree, implemented in an array).

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|

**(b)** [3 marks] Redraw the POT tree after the following operations have been carried out:
Add ”V”, Add ”F”.

**(c)** [3 marks] Redraw the tree from your previous answer after removing an element (poll).

**(d)** [10 marks] Complete the method below, which takes an array data of given size and a Comparator comp, and checks that the array is a valid Heap.

```
public boolean heapChecker(E[] data, int size, Comparator <E> comp) {












}
```

For the next four questions, consider the code for HeapSort given below, which sorts the items in the array data using a two-stage process. Both stages use pushDown, which is a *recursive* method: it finds the largest child and, if the current node is smaller, it performs a swaps and then calls pushdown on that child.

```
public void HeapSort(E[] data, int size, Comparator <E> comp) {
    for ( int i=(size-1)/2; i>=0;i--)                       // HEAPIFY
        pushDown(i,data,size,comp);
    while (size>0) {                                         // IN-PLACE DEQUEUEING
        size--;
        swap(data,size,0);
        pushDown(0,data,size,comp);
    }
}
```

**(e)** [3 marks] The loop labelled ″HEAPIFY″ begins only half-way through the array, and works back to the beginning of the array. Explain why this is so. *(Hint: It may help if you refer to the implicit tree that the array is representing).*

**(f)** [3 marks] What is the asymptotic (″big-O″) cost of the step labelled ″HEAPIFY″?

**(g)** [2 marks] What is the average-case asymptotic (″big-O″) cost of HeapSort overall?

**(h)** [2 marks] What is the worst-case asymptotic (″big-O″) cost of HeapSort overall?

## Question 7. Various topics: costs, sorting, hashing [37 marks]

The following printNums method prints out integers.

```
public void printNums(int n) {
  while (n > 0) {
    System.out.printf("%d ", n);
    n = n/2;
  }
}
```

**(a)** [2 marks] When called with printNums(20) what would the output be?

**(b)** [3 marks] What is the asymptotic ("big-O") cost of printNums($n$) as a function of $n$, and why?

The following printNumsAltered method also prints out integers.

```
public void printNumsAltered(int n) {
  while (n > 0) {
    for ( int k=0; k<n; k++)
        System.out.printf("%d ", k);
    System.out.println(" ");
    n = n/2;
  }
}
```

**(c)** [3 marks] Show the output that would be produced by printNumsAltered(10)

**(d)** [4 marks] What is the asymptotic ("big-O") cost of printNumsAltered($n$) as a function of $n$? Include a brief explanation of why this is so.

**(e)** [6 marks] Suppose you are given the list 2, 1, 20, 13, 18, 5, 19. In the box below, list the swaps carried out, and the order that this results in, when the above list is sorted by SelectionSort. Use the following format for your answers:

```
swap 2-1,  leaving  1   2  20  13  18   5  19
swap ...
swap ...
swap ...,  leaving  1   2   5  13  18  19  20
```

**(f)** [6 marks] Write code for a recursive method that makes onions with layers specified as Strings, like this:

- onion(4, ”(” , ”)” ) should produce the line ( ( ( ( ) ) ) )
- onion(4,”-”,”+”) should produce the line - - - - + + + +

```
public void onion(int layers, String str1, String str2) {








}
```

**(g)** [4 marks] What would onion(4,” ”, ”+ -”) produce?

**(h)** [2 marks] Name the two general approaches used to deal with collisions in a Hash Set.

**(i)** [3 marks] In a HashSet it is normal to increase the capacity when the array is only 70% full. Why don't we wait until it is 100% full?

**(j)** [4 marks] What is wrong with the following Hash function?

```
public int hashcode(String key) {
    int hash = 1;
    char[] characters = key.toCharArray();
    for ( int i=0; i < characters.length; i++)
        hash = hash + (int) (characters[i] * 256 * Math.random());
    return (hash % data.length); // where data is the hash table's array.
}
```

*******************************

# Appendices

### Possibly useful formulae:

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

### 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 |

### Some common exceptions.

- UnsupportedOperationException
- NullPointerException
- IndexOutOfBoundsException

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

```
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 contains(Object item);
   public boolean add(E item);                    // returns false if failed to add item
   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(int index, E element);
   public E remove(int index);                    // returns the item that was at index
   public boolean remove(Object element);         // returns true if element was removed

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);                // returns element

public 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 value removed, or null
   public boolean containsKey(K key);
   public Set<K> keySet();                   // returns set of all keys in Map
   public Collection<V> values();            // returns collection of all values
   public Set<Map.Entry<K, V>> entrySet(); // returns set of (key-value) pairs

Scanner class:
   public boolean hasNext()                  // Returns true if there is more to read
   public boolean hasNextInt()               // Returns true if the next token is an integer
   public String next()                      // Returns the next token (chars up to a space/line)
   public String nextLine()                  // Returns string of chars up to next newline
   public int nextInt ()                     // Returns the integer value of the next token
```