VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui

EXAMINATIONS — 2003

END-YEAR

COMP 103

Data Structures and Algorithms

Time Allowed: 3 Hours

Instructions: There are a total of 180 marks on this exam.
Attempt **all** questions.
Calculators may be used.
Non-electronic foreign language translation dictionaries may be used.
Write your answers in the boxes in this test paper and hand in all sheets.
There is documentation on the jds types at the end of the exam paper.

Questions	**Marks**
1. Basic Questions	[26]
2. Map Implementations	[19]
3. Programming with Collections	[15]
4. Implementing Queues with Arrays	[26]
5. Implementing Queues with Linked Lists	[26]
6. Binary Search Trees	[27]
7. Heaps	[24]
8. Tree Traversals	[17]

Question 1. Basic Questions [26 marks]

(a) [2 marks] What collection type allows duplicates, but doesn't care about the order of the items?

(b) [2 marks] What is the asymptotic cost of the binary search algorithm?

(c) [2 marks] Name a sorting algorithm with an asymptotic cost of $O(n^2)$.

(d) [2 marks] Name a sorting algorithm with an asymptotic cost of $O(n \log(n))$.

(e) [2 marks] Name two linear structures.

(f) [2 marks] Suppose we add the items "A", "B", and "C" to a stack, in that order. If we now remove an item from the stack, which items will still be on the stack?

(g) [2 marks] Suppose we add the items "A", "B", and "C" to a queue, in that order. If we now remove an item from the queue, which items will still be on the queue?

(Question 1 continued on next page)

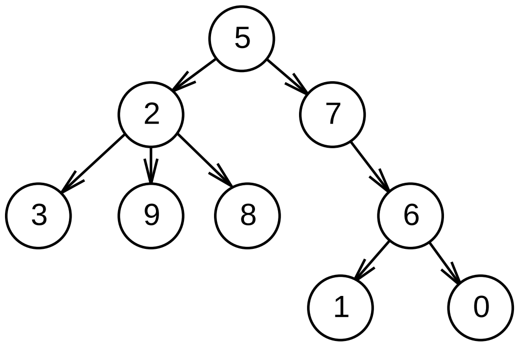
(Question 1 continued)

(h) [2 marks] If we use a Hash Table with buckets to implement the Map type, what type must the buckets be?

(i) [2 marks] If we are adding a value to a Hash Table using linear probing ("open hashing"), and the first cell we look at is already full, where do we look next?

(j) [2 marks] What is the maximum number of children a node may have in a Binary Search Tree?

Consider the following general tree:



(k) [2 marks] What value is in the root node of the tree above?

(l) [2 marks] What value is in the parent of the node with the value "6"?

(m) [2 marks] What is a value in one of the leaf nodes of the tree above?

Question 2. Map Implementations [19 marks]

This question concerns the design of a program that allows the user to look up phone numbers in a directory of company names and phone numbers. The program stores the names and numbers in a Map, using the names as the keys. The map could be implemented in several different ways.

(a) [5 marks] Assume the map contains n elements. State the asymptotic cost of searching for a key in the map if the map is implemented in each of the following ways:

Unsorted array of key–value associations:	
Sorted array of key–value associations:	
Sorted linked-list of key–value associations:	
Open addressing hash table, at most 70% full:	
Binary search tree (reasonably balanced):	

(b) [9 marks] Suppose the program must read all the names and phone numbers from a file. State the asymptotic cost of reading *all* the data from the file into a Map. Do this for each of the following implementations for the two cases when the file contains the names in sorted order and when it contains the names in random order. Assume the file contains n names.

Note: this is *not* the cost of inserting a *single* item into the map.

	file in sorted order	file in random order
Unsorted array of key–value associations:		
Open addressing hash table, at most 70% full:		
Binary search tree:		

(Question 2 continued on next page)

(Question 2 continued)

(c) [5 marks] If there are about 1,000,000 phone numbers, but users typically look up at most three phone numbers each time they run the program, explain which implementation of Map you would use for the program and justify your answer.

continued...

Question 3. Programming with Collections [15 marks]

Suppose you are designing a NameChecker program to help users assign unique names to entities (such as tropical storms or computers on a network). The program must ensure that each name is only used once.

The program stores a collection of all the names that have already been assigned. When the user wants to assign a new name, the program will first check that it hasn't been used, and then add the name to the collection. The program also lets a user cancel a name — taking it out of the collection so the name can be reused.

(a) [3 marks] Specify an appropriate type for the names field that contains the collection of names that have been assigned already.

```
public class NameChecker implements ActionListener{

    private ............................ names;
            ⋮
}
```

(b) [6 marks] Complete the following assignNewName method that reads a proposed name from the user. It then either reports (with System.out.println) that the name has already been assigned, or adds the name to the collection of assigned names and reports that the name is accepted.

```
public void assignNewName(){
    String name = JOptionPane.showInputDialog("New name:");















}
```

(Question 3 continued on next page)

(Question 3 continued)

(c) [6 marks] Complete the following cancelName method that reads a name from the user. It either reports (with System.out.println) that the name was not already assigned, or removes the name from the collection of assigned names and reports that the name has been cancelled.

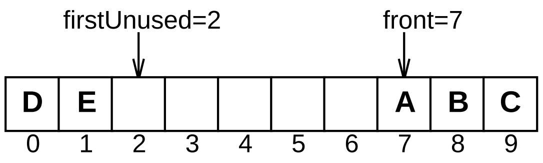
```
public void cancelName(){
    String name = JOptionPane.showInputDialog("Name to cancel:");










}
```

Question 4. Implementing Queues with Arrays [26 marks]

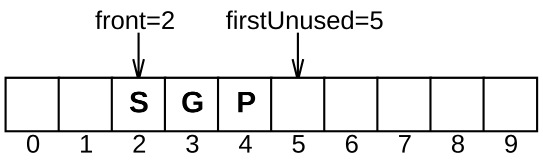
The ArrayQueue class is an implementation of Queue that holds the data items in an array. An ArrayQueue object has two indexes: one index of the cell containing the item at the front of the queue and one index of the first cell after the item at the back of the queue. Part of the code of ArrayQueue is shown below:

```
public class ArrayQueue implements Queue {
    Object[ ] data = new Object[10];
    int front = 0;
    int firstUnused = 0;

          ⋮
```

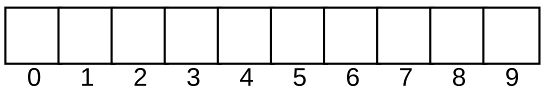
Note that in this implementation, the items in the queue can "wrap around" to the beginning of the array. For example, the following queue contains 5 items; "A" is at the front of the queue, and "E" is at the back of the queue.



(a) [3 marks] Given the following state of a queue:



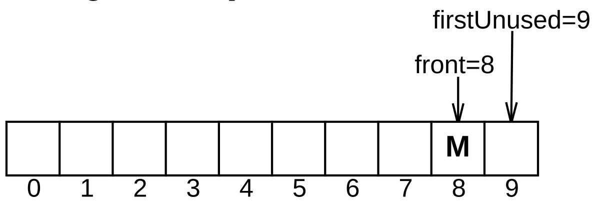
show the state of the queue after addLast("R") has been called on the queue.



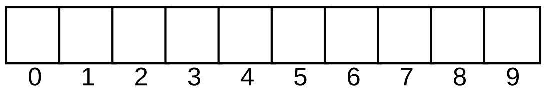
(Question 4 continued on next page)

(Question 4 continued)

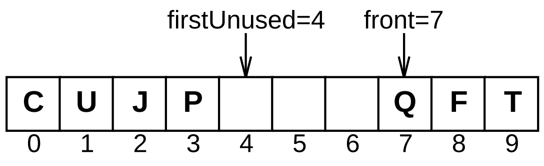
(b) [3 marks] Given the following state of a queue



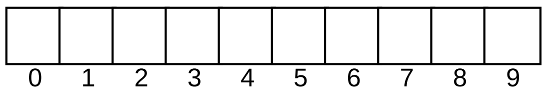
show the state of the queue after addLast("X"), addLast("Z"), and addLast("W") have been called on the queue.



(c) [3 marks] Given the following state of a queue



show the state of the queue after removeFirst(), removeFirst(), addLast("H"), and removeFirst() have been called on the queue.



(Question 4 continued on next page)

(Question 4 continued)

(d) [8 marks] Complete the following addLast method that adds an item to the back of the queue.

```
public void addLast(Object val) {
    if (val == null) throw new NoSuchElementException();
    ensureCapacity();      // ensures that there is room to add a new item









}
```

(e) [9 marks] Complete the following ensureCapacity() method, which first checks whether the data array is full. If so, it creates a new array of twice the size, and copies the items over.
Note that the fields front and firstUnused must now index the correct positions in the doubled array.

```
public void ensureCapacity() {
























}
```

continued...

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.

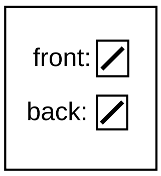
continued...

Question 5. Implementing Queues with Linked Lists [26 marks]

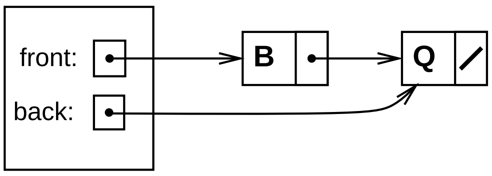
The LinkedListQueue class is an implementation of Queue that holds the data in a linked list. A LinkedListQueue object has two pointers: one to the first node in the linked list, and one to the last node in the linked list. An empty queue is represented by front and back being **null**. Part of the code of LinkedListQueue is shown below:

```
public class LinkedListQueue implements Queue {
    private ListNode front;
    private ListNode back;

          ⋮

    private class ListNode {
        public Object value;
        public ListNode next;

        public ListNode(Object v) {
            value = v;
        }
    }
}
```

An empty queue and a queue containing “B” (at the front) and “Q” (at the back) are shown below:



Empty Queue



(a) [2 marks] What are the asymptotic costs of adding and removing an element from the queue if the queue contains n items?

cost of adding:	cost of removing:

(b) [4 marks] Complete the following removeFirst method that removes the item from the front of the queue. It should throw a NoSuchElementException if the queue is empty.

```
public void removeFirst() {

}
```

(c) [8 marks] Complete the following addLast method that adds an item to the back of the queue.

```
public void addLast(Object value) {
    if (value == null) throw new NoSuchElementException();









}
```

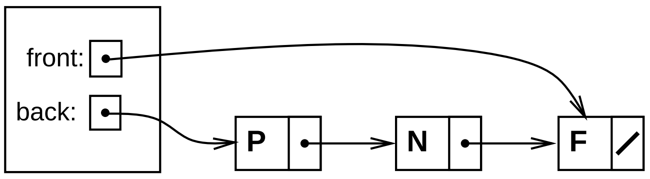
(d) [6 marks] Complete the following size method that returns the number of items in the queue.

```
public int size() {







}
```

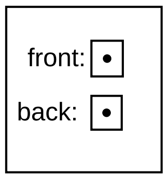
(Question 5 continued on next page)

(Question 5 continued)

Suppose that we had chosen to interpret the first node of the linked list as the back of the queue and the last node as the front of the queue, as in the following diagram:



(e) [2 marks] Using this alternative interpretation, show what state the queue above should be in after "G" is added to the back of the queue above and the item at the front of the queue is removed.



(f) [4 marks] Explain why this alternative interpretation is a bad design. Include the asymptotic costs for adding and removing an element from the queue in your answer.

SPARE PAGE FOR EXTRA ANSWERS

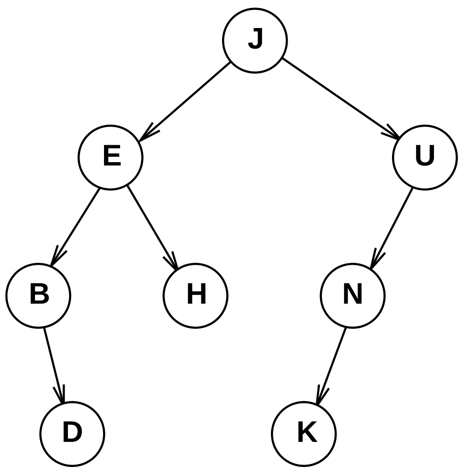
Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

continued...

Question 6. Binary Search Trees [27 marks]

(a) [4 marks] What property must be true of a binary tree for it to be a Binary Search Tree?

Consider the following Binary Search Tree containing a collection of strings:



(b) [4 marks] Draw the tree that will result from inserting the value "P" into the tree above.

(c) [2 marks] What sequence of strings in the tree would have to be compared with “P” while it was being added to the tree?

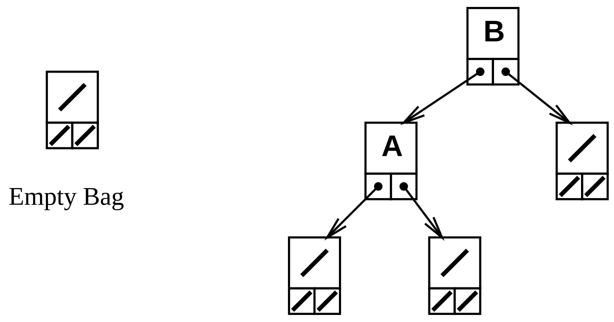
(Question 6 continued on next page)

(Question 6 continued)

The BSTBag class is an implementation of Bag using a Binary Search Tree. BSTBag uses "sentinel" nodes — every leaf of the tree is an empty node. Part of the code of BSTBag is shown below:

```
public class BSTBag implements Bag {

    private Object data;
    private BSTBag left = null;
    private BSTBag right = null;

    public BSTBag() {      // construct an empty Bag
        data = null;
    }

    public boolean isEmpty() {
        return (data == null);
    }
```

An empty tree can be represented by a BSTBag object with **null** in its data field and its left and right fields. The diagram below shows the implementation of an empty Bag and a Bag containing the two values "A" and "B".



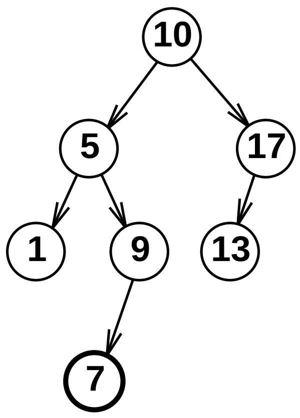
(d) [8 marks] Complete the following containsElement method that returns true if the Bag contains an item matching the argument, and false otherwise. Compare items using the compareTo method.

```
public boolean containsElement(Object val) {







}
```

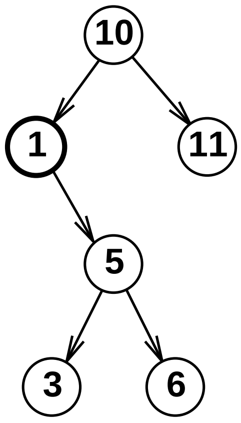
(Question 6 continued on next page)

(Question 6 continued)

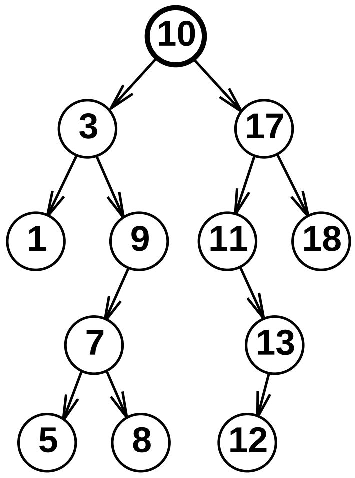
(e) [3 marks] Draw the tree that will result from removing the value "7" from the binary search tree below.



(f) [3 marks] Draw the tree that will result from removing the value "1" from the binary search tree below.



(g) [3 marks] Draw the tree that will result from removing the value "10" from the binary search tree below.



continued...

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.

continued...

Question 7. Heaps [24 marks]

A heap is a *complete binary tree* which satisfies the *partially ordered tree* property.

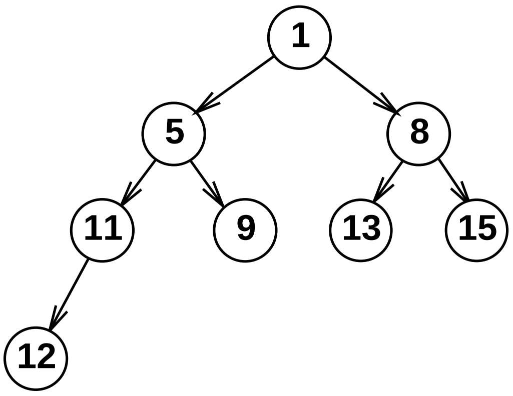
(a) [3 marks] What is meant by the term *partially ordered tree*?

(b) [3 marks] What is meant by the term *complete binary tree*?

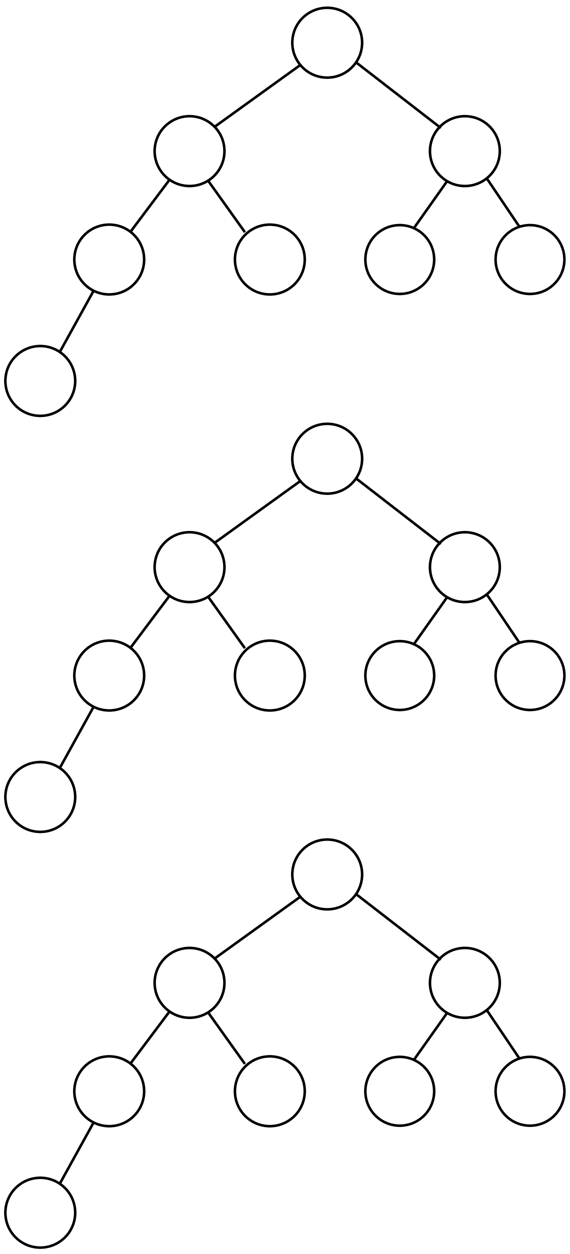
(Question 7 continued on next page)

(Question 7 continued)

(c) [5 marks]



Complete the diagrams below to show the three steps required to remove the top value ("1") from the heap shown above. (You may modify the tree structure.)



(Question 7 continued on next page)

(Question 7 continued)

Because heaps are complete binary trees, it is possible to store a heap efficiently in an array data structure, with the root stored at index 0.

(d) [3 marks] If a node is stored at index i, what are the indexes of its children.

(e) [10 marks] Suppose a Heap class has fields defined as follows:

```
public class Heap implements FindMin {
    private int elementCount = 0;
    private Object[ ] elementData = new Object[100];
    ⋮
}
```

Complete the following addElement method that adds a new item to the heap. Compare items with the compareTo method.

```
public void addElement(Object val) {
    ensureCapacity();      // ensures that there is room to add the item.












}
```

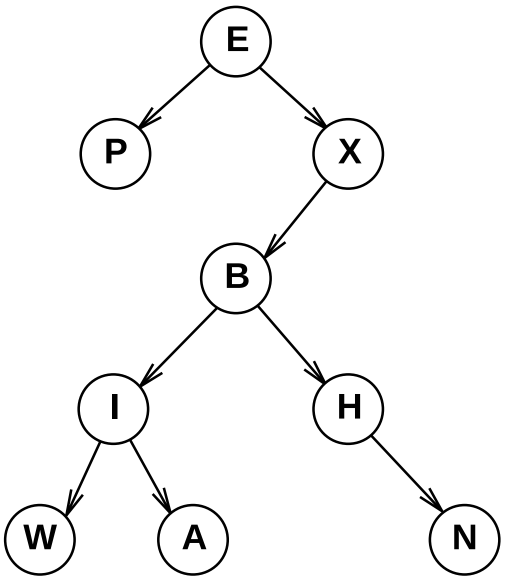
SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

continued...

Question 8. Tree Traversals [17 marks]

Consider the following tree:



Suppose a printNodes method traverses the tree and prints the value of each node as it visits the node.

(a) [3 marks] What would printNodes print out if it used an *inorder traversal*?

(b) [3 marks] What would printNodes print out if it used a *preorder traversal*?

(c) [3 marks] What would printNodes print out if it used a *postorder traversal*?

(d) [3 marks] What would printNodes print out if it used a *breadth first traversal*?

(Question 8 continued on next page)

(Question 8 continued)

(e) [5 marks] Draw a *complete* binary tree for which printNodes with an *inorder traversal* would generate the same output as your answer in part (a).

****

Appendix

Brief specifications of the jds interfaces that you may need to use in this test.

```
public interface Enumeration {
    public boolean hasMoreElements();
    public Object nextElement();
}

public interface Comparable {
    public int compareTo(Object o);
}

public interface BufferedReader{
    public String readLine();
}

public interface Collection {
    public boolean isEmpty ();
    public int size ();
    public Enumeration elements ();
}

public interface Indexed extends Collection{
    // Implementations: Vector
    public Object elementAt (int index);
    public void setElementAt (Object v, int index);
    public void addElementAt (Object val, int index);
    public void removeElementAt (int index);
}

public interface Bag extends Collection {
    // Implementations: ArrayBag, SortedArrayBag, BucketHashBag, OpenHashBag
    public void addElement (Object value);
    public boolean containsElement (Object value);
    public Object findElement (Object value);
    public void removeElement (Object value);
}

public interface Set extends Bag {
    // Implementations: ArraySet, SortedArraySet, BucketHashSet, OpenHashSet
    // operations of Bag and the following
    public void unionWith (Bag aSet);
    public void intersectWith (Bag aSet);
    public void differenceWith (Bag aSet);
    public boolean subsetOf (Bag aSet);
}
```

```
public interface Map extends Collection {
    // Implementations: ArrayMap, SortedArrayMap, BucketHashMap, OpenHashMap
    public boolean containsKey (Object key);
    public Object get (Object key);
    public void removeKey (Object key);
    public void set (Object key, Object value);
}

public interface Stack extends Collection {
    // Implementations: Vector, ......
    public void addLast (Object value);
    public Object getLast ();
    public void removeLast ();
}

public interface Queue extends Collection {
    // Implementations:
    public void addLast (Object value);
    public Object getFirst ();
    public void removeFirst ();
}

public interface FindMin extends Collection {
    // Implementations: HeapPQueue
    public void addElement (Object value);
    public Object getFirst ();
    public void removeFirst ();
}

public interface SortAlgorithm {
    // Implementations: MergeSort, InsertionSort, SelectionSort, BubbleSort, Partition
    // Constructors require a Comparator
    public void sort(Indexed collection);
}
```