VICTORIA UNIVERSITY OF WELLINGTON

Te Whare Wananga o te Upoko o te Ika a Maui

EXAMINATIONS — 2002

END-YEAR

COMP 103

Introduction to
Data Structures and Algorithms

Time Allowed: 3 Hours

Instructions:

- There are a total of 180 marks on this exam, so allocate approximately one minute per mark.
- Attempt **all** questions.
- Answer in the spaces provided if possible. If you write your answers elsewhere, indicate clearly where they are to be found.
- Non-electronic foreign language translation dictionaries are permitted.
- Non-programmable, non-alphanumeric calculators are permitted.

Questions

1. Short Questions [30 marks]
2. Bags [30 marks]
3. Queues [26 marks]
4. Trees [30 marks]
5. Heaps [30 marks]
6. Graphs [34 marks]

The following facts may be useful:

$$1 + 2 + 3 + 4 + ... + n = n(n-1)/2$$

$$1 \times 2 \times 3 \times 4 \times ... \times n = n!$$

$$1 + 2 + 4 + 8 + ... + 2^k = 2^(k+1) - 1$$

$$\text{if } 2^k = n \text{ then } k = \log_2(n)$$

Question 1. Short Questions [30 marks]

(a) [9 marks] **Bags and Asymptotic Costs.**

Complete the following table showing the average case asymptotic cost ("Big O") of the `contains`, `addElement`, and `removeElement` operations for three different implementations of `Bag` — an unsorted array, a sorted array, and a binary search tree. Express the cost in terms of n, the number of items in the bag. Assume that the binary search tree is currently balanced.

`Bag` Implementation

Operation	Unsorted Array	Sorted Array	Binary Search Tree
`contains`			
`addElement`			
`removeElement`			

(b) [4 marks] **Hash Tables: Collisions.**

Hash tables with buckets and hash tables with open addressing differ in what they do when an item hashes to a position in the array that already contains a different item. Briefly explain the difference.

Hash tables with buckets have a collection of item at each position in the array. If an item hashes to a position that already contains an item, the new item is simply added to the collection at that position.
Hash tables with open addressing have only a single item at each position in the array. If an item hashes to a position that already contains an item, the hash table will "probe" other locations until it finds an empty location to hold the item.

(c) [4 marks] **Hash Tables: Probing.**

Linear and quadratic probing are two different methods for probing in hash tables. Briefly explain each method in terms of how they step to the next location, and explain why quadratic probing is better than linear probing.

Linear probing always steps by 1 to the adjacent location in the array. Quadratic probing steps by increasing amounts, where the steps increase quadratically (*i.e.*, the size of the step is the square of the number of the step).
Quadratic probing is better because the "runs" from different starting positions do not combine into very long runs.

(Question 1 continued on next page)

(Question 1 continued)

(d) [8 marks] **Loops and Asymptotic Costs.**

In the code fragments below, `data` is an array of `int`. Assume the length of `data` is n.

What is the asymptotic cost of the following loop, expressed in terms of n?

```
for (int i=0; i < data.length; i++){
    for (int j=0; j < data.length; j++){
        data[i]++;
    }
}
```

$O(n^2)$

What is the asymptotic cost of the following loop, expressed in terms of n?
(Note the difference in the first `for` loop.)

```
for (int i=1; i<data.length; i=i*2){
    for (int j=0; j<i; j++){
        data[i]++;
    }
}
```

$O(n)$

What is the asymptotic cost of the following loop, expressed in terms of n?

```
int lo = 0;
int hi = data.length-1;
while (lo < hi){
    int mid = (lo + hi)/2;
    if ( data[mid] > 0 )
        hi = mid-1;
    else
        lo = mid+1;
}
```

$O(\log_2 n)$

(e) [5 marks] **Sorting Algorithms.**

For each of the following sorting algorithms, give its average case asymptotic cost, in terms of n (the number of items to be sorted).

Bubble Sort	
Heap Sort	
Insertion Sort	
Merge Sort	
Quicksort	
Selection Sort	
Shell Sort	
Tree Sort	

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 2. Bags [30 marks]

A `Bag` is a collection of items that allows duplicates. It attaches no meaning to the order in which items are stored.

An item may occur a large number of times in a `Bag`, but the `Bag` interface does not provide a method to count how many occurrences of an item there are.

Consider the following method that uses the methods in the `Bag` interface to determine the number of occurrences of an item in a `Bag`. It repeatedly removes an item from the `Bag`, counting as it goes, until the item is no longer present. It then adds the item to the `Bag` the same number of times.

```
public static int occurrences(Object item, Bag b){
    int count = 0;
    while (b.containsElement(item)){
        b.removeElement(item);
        count++;
    }
    for (int i=0; i<count; i++){
        b.addElement(item);
    }
    return count;
}
```

(a) [3 marks] Suppose a `Bag` is implemented with an unsorted array (for example, an `ArrayBag`). If the `Bag` contains n items in total and there are at most m occurrences of any item, what is the worst case asymptotic cost of the `occurrences` method, expressed in terms of n and m?

Cost of removing all m occurrences: $O(mn)$

Cost of replacing them: $O(m)$

Total cost: $O(mn)$

(b) [3 marks] Suppose a `Bag` is implemented with a sorted array (for example, a `SortedVector`). If the `Bag` contains n items in total and there are at most m occurrences of any item, what is the worst case asymptotic cost of the `occurrences` method, expressed in terms of n and m?

Cost of removing all m occurrences: $O(mn)$

Cost of replacing them: $O(mn)$

Total cost: $O(mn)$

(c) [4 marks] The `occurrences` method modifies the `Bag` while it is counting the number of occurrences of an item. At the end of the `occurrences` method, will the `Bag` be different from its previous state in any way that matters? Explain why or why not.

The order of items in the bag may be changed, but a `Bag` makes no commitment about the order of items, so this does not matter. [3]

However, the method removes all items in the bag that are `equal` to the given item, and replaces them with identical copies of that item. If the items are equal but do not contain the same information (for example, if they are association pairs and have equal keys but different values, then the bag will have changed in a way that matters. [1]

(d) [5 marks] A more efficient way of counting the number of occurrences of an item in a `Bag` would be to iterate through all the elements using the `Enumeration` returned by the `elements` method, and count how many elements are equal to the given item. Complete the following implementation of the `occurrences` method using this approach.

```
public static int occurrences(Object item, Bag b){
  int count = 0;
  for(Enumeration e  = b.elements(); e.hasMoreElements();){
    if (item.equals(e.nextElement()))
      count++;
  }
  return count;









}
```

(e) [2 marks] What is the asymptotic cost of your version of `occurrences` (in part **(d)**) in terms of n and m?

$O(n)$

(Question 2 continued on next page)

(Question 2 continued)

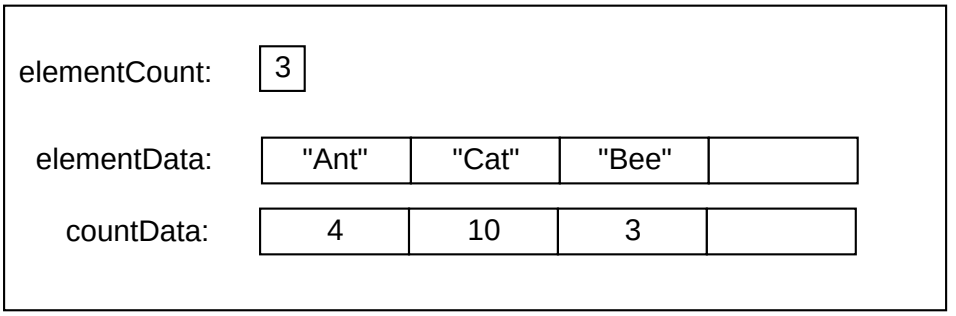
If a `Bag` contains a large number of duplicate items, it would be more efficient (in both space and time) to store each distinct item just once, along with a count of how many times it occurs in the `Bag`, than to store each item many times. One way of doing this is to have two arrays (of the same size), one to store the distinct items and one to store the counts, as in the following `ArrayCountBag` class.

```
public class ArrayCountBag implements Bag {
  private int elementCount = 0;            // number of distinct items
  private Object[] elementData = new Object[10]; // the data items
  private int[] countData = new int[10];         // the counts

  ⋮

}
```

For example, an `ArrayCountBag` containing 4 copies of the string `"Ant"`, 3 copies of the string `"Bee"`, and 10 copies of the string `"Cat"` might look like this:



(f) [5 marks] Complete the following `size` method which should return the total number of items in a `Bag`. To compute the number of items, it will need to add up all the counts of the distinct items.

```
public int size () {
  int count = 0;
  for (int i=0; i<elementCount; i++){
    count += countData[i];
  }
  return count;
}




}
```

(g) [8 marks] Complete the following `addElement` method which adds an item to a `Bag`. If the `Bag` already contains a copy of the item, it should increment the count of the item. Otherwise, it must add the item and set its count to 1.

```
public void addElement (Object item) {
  for (int i=0; i<elementCount; i++){
    if (item.equals(elementData[i])){
      countData[i]++;
      return;
    }
  }




  ensureCapacity();  // ensure the arrays are large enough to hold new values
  elementData[elementCount]=item;
  countData[elementCount]=1;
  elementCount++;



}
```

Question 3. Queues [26 marks]

The following is a simple implementation of Budd's `Queue` interface, using his `Vector` class to hold the queue elements:

```
class VectorQueue implements Queue {

  private Vector data = new Vector();
  private int elementCount = 0;

  public VectorQueue () { };

  public boolean isEmpty () {
    return elementCount == 0;
  }

  public int size () {
    return elementCount;
  }

  public void addLast (Object val) {
    data.addElementAt(val, elementCount);
    elementCount++;
  }

  public Object getFirst () {
    return data.elementAt(0);
  }

  public void removeFirst () {
    data.removeElementAt(0);
    elementCount--;
  }

  public Enumeration elements () {
    // Omitted
  }

}
```

(a) [12 marks] **Analysis.**

Give the asymptotic cost of each of the following operations (in the implementation shown opposite), for a queue containing n elements. Briefly justify each answer.

`size`:

$O(1)$. Size is stored, so just need to return its value.

`addLast`:

$O(1)$. Adding an element to a vector has constant cost (averaged over many operations).

`getFirst`:

$O(1)$. Accessing an element of a vector has constant cost.

`removeFirst`:

$O(n)$. To remove the first element of a vector, you have to move all the other elements down one, which has linear cost.

(b) [4 marks] **Analysis.**

Explain, with reference to your answer to part **(a)** above, why this is **not** a good implementation for queues.

With this implementation, `removeFirst` has linear cost, whereas there are other implementations (such as using a linked list) for which all of the above operations have constant cost.

(Question 3 continued on next page)

(Question 3 continued)

(c) [10 marks] **Improving the implementation.**

On the following copy of the `VectorQueue` class, show the changes needed to obtain a more efficient implementation of queues.

Add comments to the right of the code explaining your changes.

The new implementation should still use a `Vector` to hold queue elements, and only needs one or two additional variables.

Note: To get full marks, your implementation must be able to use all of the locations in the `Vector` to store queue elements. You can get 8 of the 10 marks for a version that does not do this.

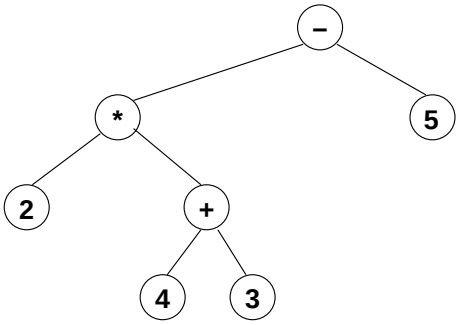
```
  public Enumeration elements () {
    // Omitted
  }
}
```

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. Trees [30 marks]

We can use a binary tree to represent arithmetic expressions composed of binary operators (i.e. operators that take two arguments) and numbers. The leaves of the tree are labelled with numbers, and the non-leaf nodes are labelled with operators. We will call this kind of tree an *expression tree*. For example, the following expression tree represents the arithmetic expression `2*(4+3)-5`:



(a) [3 marks] **Counting nodes.**

If an expression tree has n nodes, how many of them will be labelled with numbers?

$\lceil n/2 \rceil$ or $(n+1)/2$

(b) [5 marks] **Pre-Order Traversal.**

Write the output that would result from a *pre-order traversal* of the above expression tree, printing the label at each node as it is visited.

`-*2+435`

Can pre-order traversals of two different expression trees produce the same result? If so, draw another expression tree for which a pre-order traversal would produce the same result as the above tree.

No.

(c) [5 marks] **Post-Order Traversal.**

Write the output that would result from a *post-order traversal* of the above expression tree, printing the label at each node as it is visited.

```
243+*5-
```

Can post-order traversals of two different expression trees produce the same result? If so, draw another expression tree for which a post-order traversal would produce the same result as the above tree.

No.

(d) [5 marks] **In-Order Traversal.**

Write the output that would result from a *in-order traversal* of the above expression tree, printing the label at each node as it is visited.

```
2*4+3-5
```

Can in-order traversals of two different expression trees produce the same result? If so, draw another expression tree for which an in-order traversal would produce the same result as the above tree.

Yes. (There are lots - add one later.)

(Question 4 continued on next page)

(Question 4 continued)

(e) [12 marks] **Evaluating Expressions.**

Write an algorithm (in pseudo-code or Java) to find the value of an arithmetic expression represented as expression tree. Assume that the only operators used are `+`, `-`, `*` and `/`. You do not need to explain the data types you use, so long as they and the names of their fields and methods are similar to those used during the course.

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 5. Heaps [30 marks]

A *heap* is a complete, partially ordered tree. A heap can be stored in an array, so that the root is at location 0, and the children of the node at location k (if they exist) are at locations $2k + 1$ and $2k + 2$. This representation of heaps provides the basis for an efficient implementation of priority queues, and for heap sort.

(a) [8 marks] **Understanding heaps.**

What is the defining property of a partially ordered tree?

For every subtree, the value at the root is "smaller than" (i.e. precedes in the appropriate ordering) the values at its children.
(Smaller than or equal to if we allow duplicates.)

What is the defining property of a complete binary tree?

Every level is "full" (i.e. has the maximum possible number of nodes), except the last level where all nodes are as far to the left as possible.

(b) [4 marks] **Implementing heaps.**

Consider the heap represented by the following array:

3	10	4	12	15	8	7	25	14	16

Draw the heap as a tree.

(Later)

(c) [9 marks] **Inserting in a heap.**

Suppose 5 is inserted in the heap given in part **(b)**. Show the resulting heap, as a tree, and its representation as an array.

(Later)

What is the asymptotic cost of inserting an element in a heap (represented as an array) containing n elements? Briefly justify your answer.

$log_2\ n$

(d) [9 marks] **Deleting in a heap.**

Suppose the root is removed from the heap given in part **(b)**. Show the resulting heap, as a tree, and its representation as an array.

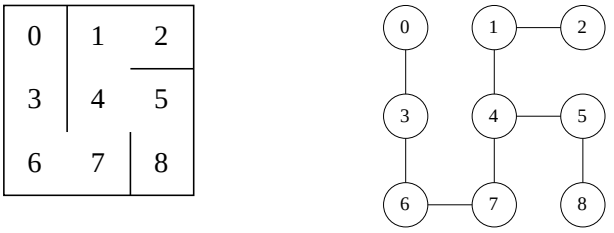
(Later)

What is the asymptotic cost of deleting the root of a heap (represented as an array) containing n elements? Briefly justify your answer.

$log_2\ n$

Question 6. Graphs [34 marks]

A maze can be represented using an undirected graph. For example, below is a very simple 3×3 maze, and a corresponding graph. Cells in the maze are numbered across successive rows, and these numbers are used to identify the corresponding nodes in the graph.



(a) [10 marks] **Representing graphs with adjacency matrices.**

Explain briefly how an *adjacency matrix* can be used to represent an undirected graph.

An adjacency matrix has one row and column for each node in the graph. The (i, j) element of the matrix (i.e. the element in row i, column j) is 1 (true) if there is an edge from node i to node j, and 0 (false) otherwise. Since the graph is undirected, the graph is symmetrical about the leading diagonal, so there is a 1 in the (i, j) element iff there is a 1 in the (j, i) element.

Show the adjacency matrix representation for the above graph (you may omit matrix entries where there is no edge).

	0	1	2	3	4	5	6	7	8
0				1					
1			1		1				
2		1				1			
3	1								
4		1						1	
5			1		1				1
6				1				1	
7					1		1		
8						1			

How many array locations are needed for an adjacency matrix representation of an $N \times N$ maze? Explain your answer. (You may assume that each node has, on average, two neighbours.)

N^4. The maze has N^2 cells, and the array has one row and one column for each cell in the maze. This is independent of the number of neighbours each node has.

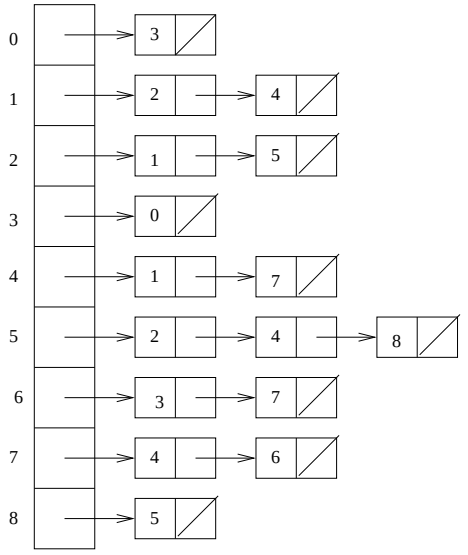
(b) [10 marks] **Representing graphs with adjacency lists.**

Explain briefly how an *adjacency list* can be used to represent an undirected graph.

An adjacency list representation has an array of lists, one for each node, giving the neighbours of that node.

Show the adjacency list representation for the above graph.

.5



How many storage locations are needed for an adjacency list representation of an $N \times N$ maze? Explain your answer.

(Assume that the adjacency list is implemented using singly linked lists, and count storage locations used to store pointers and node numbers. You may assume that each node has, on average, two neighbours.)

$5N^2$. The maze has N^2 cells, and for each cell there is an element in the main array, and on average two list nodes each of which stores a value (a cell number) and a pointer, making 5 locations for each cell.

(Question 6 continued on next page)

(Question 6 continued)

(c) [6 marks] **Finding isolated nodes.**

A graph node is *isolated* if it has no neighbours.

The following is a pseudo-code algorithm to list the isolated nodes in a graph, G:

```
for each node, x, in G:
    determine whether x has any neighbours;
    if not, print x
```

What is the worst case asymptotic cost of this algorithm for a graph with n nodes, represented using an adjacency matrix? Briefly explain your answer.

$O(n^2)$. In the worst case, we have to scan the entire row of the adjacency matrix for each node. (We can stop scanning each row as soon as we see a non-zero value, so in the best case, when the graph is complete, the cost is linear. The average cost will depend on assumptions about the distribution of edges in the graph.)

What is the worst case asymptotic cost of this algorithm for a graph with N nodes, represented using an adjacency list? Briefly explain your answer.

$O(n)$. For each node, we only need to determine whether its list of edges is empty, which has constant cost.

(d) [8 marks] **Searching.**

Write an algorithm (in pseudo-code or Java) to search for a solution to a maze represented as a graph.

The search should start at node `startNode` and stop when it finds node `goalNode` or it has examined every node, and should print a message saying whether the goal node was found or not. You do not need to explain the data types you use, so long as they and the names of their fields and methods are similar to those used during the course.

FIX!!!
Since we want the shortest distance to each node, we need to do a breadth-first traversal. To find distances, we need to store the distance of each node on the queue. So queue elements are $(node, distance)$ pairs.

```
Traverse(start):
    Add (start, 0) to queue
    Unmark all nodes
    WHILE queue not empty DO
        Let (n,k) be first pair on queue and remove it
        IF n is unmarked THEN
            Mark n and print n, k
            FOR each unmarked neighbour m of n DO
                Add (n, k+1) to queue
            END
        END
    END
```

****