TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS — 2004

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 java and jds interfaces at the end of the exam paper.

Questions	**Marks**
1. Basic Questions	[28]
2. Programming with Collections	[16]
3. Implementing Linked Lists	[20]
4. Using Stacks	[8]
5. Implementing Queues	[22]
6. Trees	[32]
7. Priority Queues	[28]
8. Graphs	[26]

Student ID: .....

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.

Student ID: .....

Question 1. Basic Questions [28 marks]

(a) [2 marks] Which collection type keeps items in order, but only allows you to add and remove elements from one end?

A Stack

(b) [2 marks] What is the asymptotic cost of searching for an item in a Bag implemented using an unsorted array?

$O(n)$

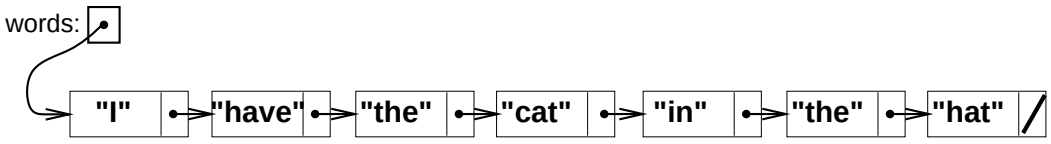
(c) [2 marks] Name a sorting algorithm with an asymptotic cost of $O(n \log(n))$.

Merge sort or Quick sort, or Heap sort

(d) [2 marks] What is the asymptotic cost of adding an item to a set implemented using a bucket hash table, where the buckets are unsorted ArraySets.

O(1)

Consider the diagram below of a variable words containing the reference of a linked list. Assume that each node of the list has the fields value and next.



(e) [2 marks] What is the value of words.value?

"I"

(f) [2 marks] What is the value of words.next.next.value?

"the

Student ID: .....

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

"B" and "C"

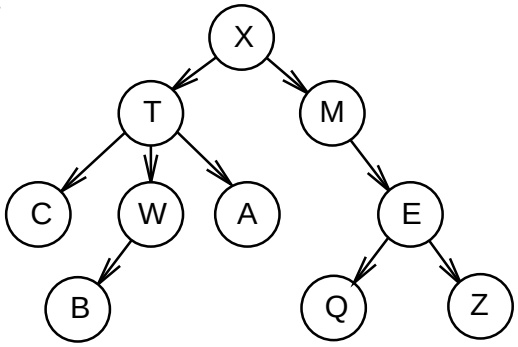
(h) [2 marks] If we are adding a value to a Hash Table using linear probing ("open hashing"), and the first cell we look at is already full, where do we look next?

The adjacent cell

(i) [2 marks] What is the maximum number of nodes that could be in a binary tree of depth 2 (three levels of nodes)

7

Consider the following general tree:



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

5

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

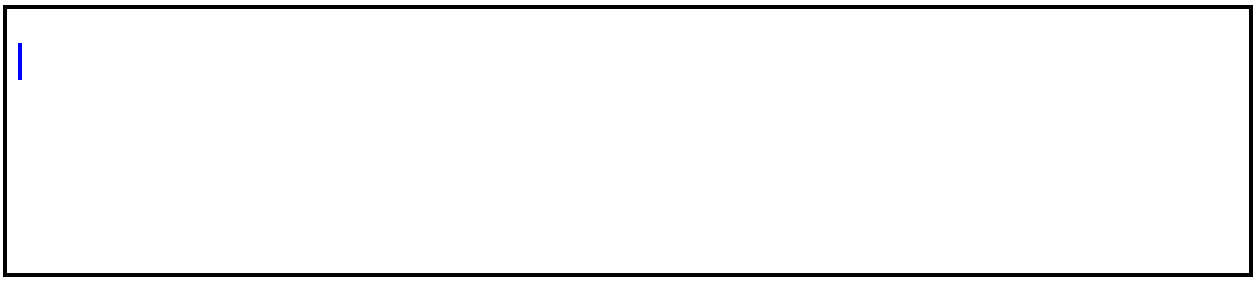
"M"

(l) [2 marks] How many leaf nodes does the tree have?

Student ID: .....

5

(m) [2 marks] Draw a graph containing four nodes and five edges.



(n) [2 marks] What is the difference between a directed graph and an undirected graph?

In a directed graph, the edges have an arrow,
representing a connection in only one direction between the two nodes.
In an undirected graph, the edges have no arrow, and the edge
represents a bidirection connection between the nodes

Student ID: .....

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.

Student ID: .....

Question 2. Programming with Collections [16 marks]

Suppose you are designing a Translator program that provides very simple translation of sentences from English to French. The Translator class contains a Map of associations between English words and French words. Keys to the map are English words, and the values are the French words.

```
public class Translator {
    private Map dictionary;        //keys are English words, values are French words.
```

(a) [6 marks] Complete the following printSentence method, whose argument is an Indexed collection of words (Strings), and prints out the sentence on a single line. Your code should use an Enumeration. You may use System.out.print and System.out.println to print to the output.

```
public static void printSentence(Indexed sentence){
    for (Enumeration e = sentence.elements(); e.hasMoreElements();)
        System.out.print(e.nextElement()+" ");
    System.out.println();



}
```

(b) [10 marks] Complete the following translate method whose argument is the English sentence to translate, represented as an Indexed collection of words, and returns a French sentence, also represented as an Indexed collection of words. It should look up each English word in the dictionary to find the equivalent French word, and put the French word into the French sentence. If there is no French translation for an English word in the dictionary, it should put the English word into the sentence instead.

```
public Indexed translate(Indexed english){
    //construct and return a sentence of french words using dictionary.
    Vector french = new Vector();
    for (int i = 0; i < english.size(); i++){
        Object word = english.elementAt(i);
        if (dictionary.containsKey(word))
            french.addElementAt(dictionary.get(word), i);
        else
            french.addElementAt(word, i);
    }




    return french;
}
```

Student ID: .....

Question 3. Implementing Linked Lists [20 marks]

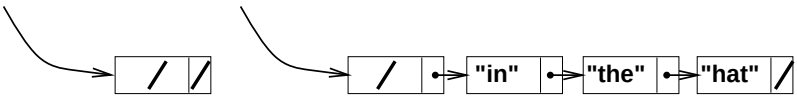
Suppose you want to implement a Collection using a linked list.

(a) [3 marks] State a problem that arises if we implement the Collection using a "lisp-style" linked list with no header and where the empty collection is represented by **null**.

You cannot call any methods on an empty collection, OR
It is hard to remove the first value in the collection

Suppose you decide to use lists with no header but with a dummy node – where every list has a node at the front that contains the **null** value, which is not part of the collection.

For example, here is an empty Collection, and a Collection with three items.



The DLList class contains the following fields and constructors:

```
public class DLList implements Collection{

    private Object value;
    private DLList next;

    /* Create an empty DLList collection */
    public DLList(){
        value = null;
        next = null;
    }

    /* Create a new node. */
    private DLList(Object val, DLList nx){
        value = val;
        next = nx;
    }
```

(b) [2 marks] Draw two diagrams below to show the structure of the collections if the element "cat" were added as the first element to each of the DLLists in the diagrams above.

fig-linkedlist-dummy-add.eps

Student ID: .....

(Question 3 continued)

(c) [4 marks] Complete the following addFirst method that adds a new value as the first element in a DLList.

```
public void addFirst (Object val) {
    if (val == null) throw new NoSuchElementException();
    next = new DLList(val, next);


}
```

(d) [5 marks] Complete the following size method that returns the number of elements in a DLList:

```
public int size () {
    if (next == null)
        return 0;
    else
        return 1 + next.size();


}
```

(e) [6 marks] Complete the following removeElement method that removes the element that equals its argument from the DLList. If the DLList is empty, it should throw a NoSuchElementException.

```
public void removeElement (Object val){
    if (val == null) throw new NoSuchElementException();
    if (next == null) throw new NoSuchElementException();
    if (val.equals(next.value))
        next = next.next;
    else
        next.removeElement(val)


}
```

Student ID: .....

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.

Student ID: .....

Question 4. Using Stacks [8 marks]

Every Collection in the jds library has an elements method that returns an Enumeration of the elements which generates the elements in some order.

If we wanted to print the elements in the reverse of this order, we could use a Stack. Complete the following printCollection method that prints the elements of a collection in the reverse of the order of the Enumeration returned by the elements method.

You should use the System.out.println(...) method to print each element.

```
public void printCollection(Collection data) {
    Stack st = new LinkedListStack();
    Enumeration e = data.elements();
    while (e.hasMoreElements()){
        st.addLast(e.nextElement())
    }
    while (! st.isEmpty()){
        System.out.println(st.getLast());
        st.removeLast();
    }

}
```

Student ID: .....

Question 5. Implementing Queues [22 marks]

The ArrayQueue class is a queue that holds the data items in an array. An ArrayQueue object has two indexes: the index of the cell containing the item at the front of the queue and the index of the first cell after the item at the back of the queue.

The code for ArrayQueue is shown below:

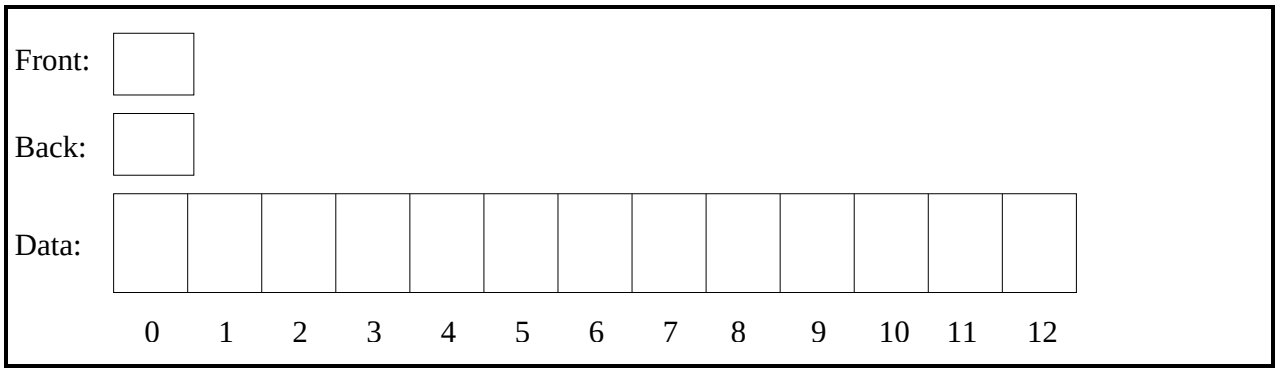
```
public class ArrayQueue {

    private Object[ ] data;
    private int front;
    private int back;

    public ArrayQueue() {
        this.data = new Object[13];
        this.front = 0;
        this.back = 0;
    }

    public void enqueue(Object val) {
        if (this.isFull())
            throw new RuntimeException("Queue Full.");
        this.data[this.back++] = val;
    }

    public Object dequeue() {
        if (this.isEmpty())
            throws new RuntimeException("Queue Empty.");
        return this.data[this.front++];
    }

    public boolean isFull() {
        return (this.back == this.data.length);
    }

    public boolean isEmpty() {
        return (this.front == this.back);
    }
}
```

(a) [6 marks] Suppose a new queue is created, and then the following seven operations are performed:

```
enqueue("A"),  enqueue("B"),  dequeue(),
enqueue("C"),  enqueue("D"),  dequeue(),  dequeue().
```

Show the resulting state of the queue:

Student ID: .....



Student ID: .....

(Question 5 continued)

(b) [6 marks] Explain the limitations of this implementation of a queue.
Hint: Look at the code for the four methods on the facing page.

It will run out of space after adding just 13 items, even if it has dequeued them all
Also, it is not able to increase the space available when the array is full

(c) [6 marks] The lectures showed a better array-based implementation for queues. Write the code for the enqueue and isFull methods for the implementation described in lectures.

```
public void enqueue(Object val) {
    if (this.isFull()) {
        throw new RuntimeException("Queue Full.");
    }
    this.data[this.back++] = val;
    if (back >= data.length)
        back = 0;


}
```

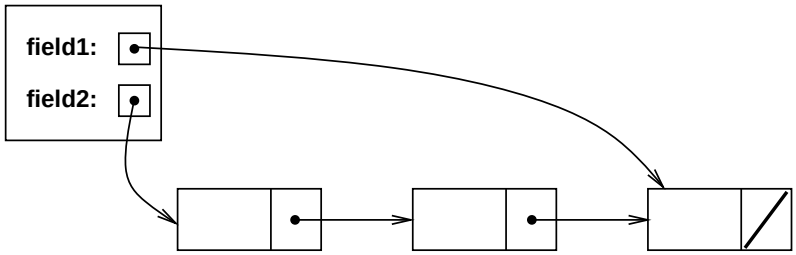
```
public boolean isFull() {
    if (back == data.length–1)
        return (front == 0);
    else
        return (front == back+1)


}
```

(Question 5 continued on next page)

Student ID: .....

(Question 5 continued)

(d) [4 marks] Suppose you decided to implement a queue using a linked-list data structure with a header containing two fields with pointers to the nodes at each end of the linked-list, as shown below:



Which field would you use for the back of the queue (where items are enqueued) and which field would you use for the front of the queue (where items are dequeued from)? Briefly explain your choice.

Front: field 2

Back: field 1

It is cheap to add a node after the node that field 1 points at,
but it is expensive to delete that node, so this should be where we add nodes
It is cheap to add or delete nodes at field 2

Student ID: .....

Question 6. Trees [32 marks]

(a) [3 marks] What property must a binary tree have to be a binary search tree?

For every node in the tree, all the values in the left subtree must be less than or equal to the value in the node, and all the values in the right subtree must be greater than or equal to the value in the node

(b) [3 marks] What property must be true of a binary search tree (BST) for insertion/access/deletion to have O($log(n)$) complexity in all cases?

It must be balanced, (all leaves must be in the bottom two levels of the tree)

(c) [2 marks] Show the resulting tree when the following items are inserted (in this order) into an empty binary search tree:

"A", "B", "C", "D", "E", "F", "G", "H"

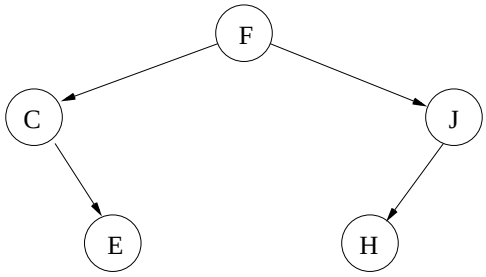
(Question 6 continued on next page)

Student ID: .....

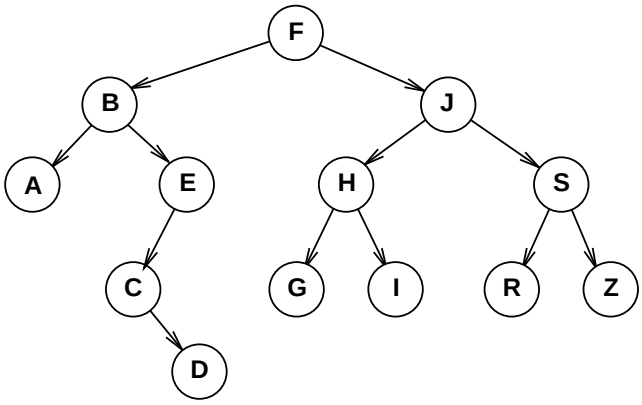
(Question 6 continued)

(d) [4 marks] Show the resulting tree when the following items are inserted (in this order) into the binary search tree below:

"P", "L", "G", "I", "D", "B"



(e) [4 marks] Consider the following binary search tree:



State the order that the values of this tree would be visited during a post-order, depth-first traversal of the tree.

A, D, C, E, B, G, I, H, R, Z, S, J, F

(f) [5 marks] The breadth-first traversal algorithm uses a collection to store the nodes that are to be visited. This collection keeps the nodes in the order that they are to be visited. Draw the state of this collection during a breath-first traversal of the binary tree above at the point when node "C" has just been added to the collection.

front of queue: H, S, C, :back of queue

Student ID: .....

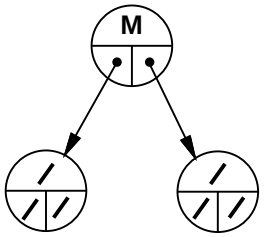
(Question 6 continued on next page)

Student ID: .....

(Question 6 continued)

(g) [5 marks] Suppose the TreeNode class below is an implementation of a binary tree using dummy nodes for the children of leaf nodes. The diagram shows a tree containing a single element.

```
public class TreeNode {

    private TreeNode left;
    private TreeNode right;
    private Object data;

    /* Constructor of a dummy node (an empty tree) */
    public TreeNode() {
        this.left = null;
        this.right = null;
        this.data = null;
    }
        ⋮
```



Complete the following size method for the TreeNode class that should return the number of elements in a binary tree.
Hint: use a recursive method.

```
public int size() {




}
```

(h) [3 marks] The TreeSort sorting algorithm inserts data into a BST one item at a time, and then traverses the BST using an in-order depth-first tree traversal to construct the sorted list. What is the cost of this TreeSort algorithm (using $O()$ notation) if the data to be sorted is already sorted?

$O(n^2)$

(i) [3 marks] If we use a heap to implement a Priority Queue, we get O($log(n)$) performance for both *enqueue* and *dequeue* in all cases. Suppose we used a heap to implement a Bag. Would we still be able to get O($log(n)$) performance for *addElement* and *removeElement* in all cases? If so, why? If not, why not?

No, because searching for an item would take $O(n)$ steps, since we might have to search the whole tree

Student ID: .....

Question 7. Priority Queues [28 marks]

Consider the following two implementations of a priority queue.

Implementation 1: An array of queues, with the array indexed by the priority level, and the queues having O(1) complexity for both enqueue and dequeue operations.

Implementation 2: A heap, with O(log(n)) complexity for both enqueue and dequeue operations.

(a) [4 marks] Which implementation is better when the number of possible priority levels is small? Explain why.

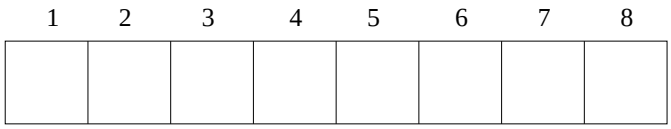
implementation 1, because it is O(1),

(b) [4 marks] Which implementation is better when the number of possible priority levels is large? Explain why.

implementation 2, because implementation 1 would require too large an array

(c) [3 marks] Suppose a priority queue is implemented as an array of queues. Draw the priority queue after the following data has been inserted into an empty priority queue.
(The letter is the data, and the number in brackets is the priority.)

Data Items: A (1), B (5), C (3), T (4), Y (2), R (5), H (2), S (3)



(d) [3 marks] List the values in the order they would be dequeued from the priority queue in question (c).

A, Y, H, C, S, T, B, R

(Question 7 continued on next page)

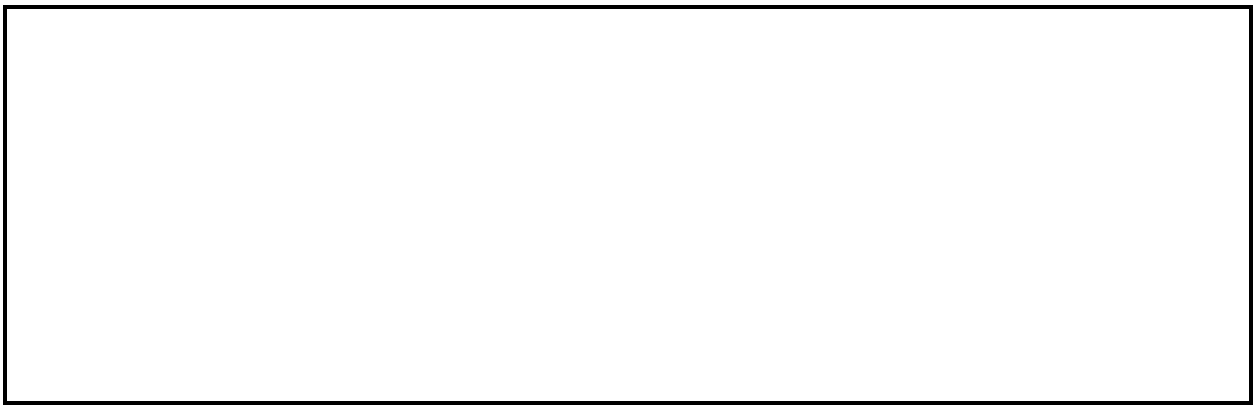
(Question 7 continued)

(e) [4 marks]

Suppose a priority queue is implemented as a heap. Draw the heap after the following data items have been inserted. The letter is the value, and the number in brackets is the priority.

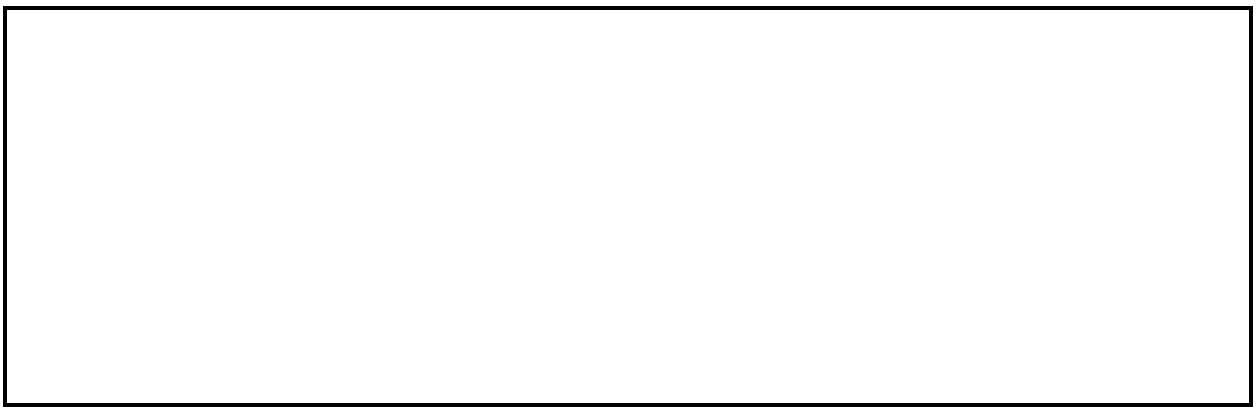
Data Items: A (1), B (5), C (3), T (4)

Hint: draw this heap as a tree rather than as an array.



(f) [3 marks] Draw the new heap after the following data items have been inserted into the heap you created in question (e).

Data Items: Y (2), R (5), H (2), S (3)



(Question 7 continued on next page)

Student ID: .....

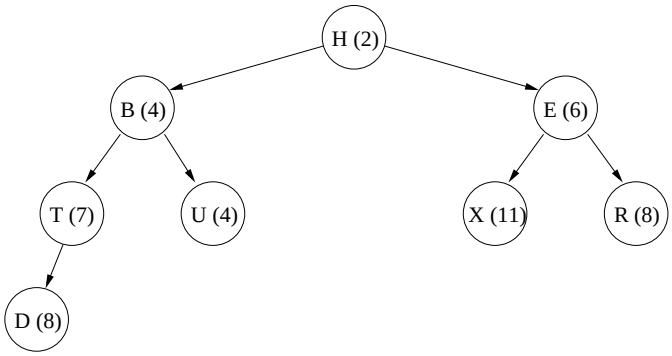
SPARE PAGE FOR EXTRA ANSWERS

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

Student ID: . . . . . .

(Question 7 continued)

Suppose a heap contains the following items:



(g) [4 marks]

Draw the new heap after *dequeue()* has been called two times on the heap above.

(h) [3 marks]

Draw the heap after *dequeue()* has been called another two times on the heap created in question (g).

Student ID: .....

Question 8. Graphs [26 marks]

(a) [6 marks] Draw the graph that is represented by the following undirected, weighted adjacency matrix. Assume that a blank square represents no edge.

	A	B	C	D	E	F	G
A			4				
B			1			3	
C	4	1		6		2	4
D			6				
E							
F		3	2				
G			4				2

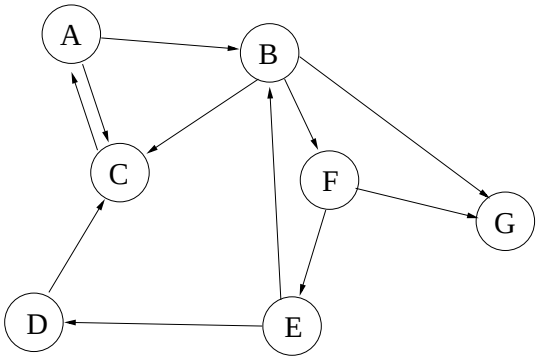
Graph:

(Question 8 continued on next page)

Student ID: .....

(Question 8 continued)

(b) [6 marks] Draw an adjacency list representation of the following directed, unweighted graph.



(c) [5 marks] State one reason why a tree traversal algorithm might not work on a directed graph.

Loops/cycles,
sources (nodes that have no edges into them)

(d) [5 marks] To turn a tree traversal algorithm into a graph traversal algorithm, one change you would usually make is to use an additional Set data structure. What would this Set contain at any given point during a graph traversal?

The nodes that have already been visited during the traversal

(e) [4 marks] Give an example of a directed, weighted graph where a breadth-first traversal would not find the least-cost path.

Student ID: .....

****

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);
}
```