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

EXAMINATIONS — 2006

SUMMER TRIMESTER

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 1 minute per mark.
4. Write your answers in the boxes in this test paper and hand in all sheets.
5. Non-electronic foreign language-English translation dictionaries are permitted.
6. Calculators are **not** permitted.
7. There is documentation on the relevant Java classes and interfaces at the end of the exam paper.

Questions	**Marks**
1. Basic Questions	[18]
2. Sorting	[20]
3. Analysis of Algorithms	[15]
4. Programming with Collections	[15]
5. Binary Search Trees	[25]
6. General Trees	[18]
7. Heaps	[23]
8. Priority Queues	[18]
9. Hashing	[18]
10. Graphs	[10]

Student ID: .....

Question 1. Basic Questions [18 marks]

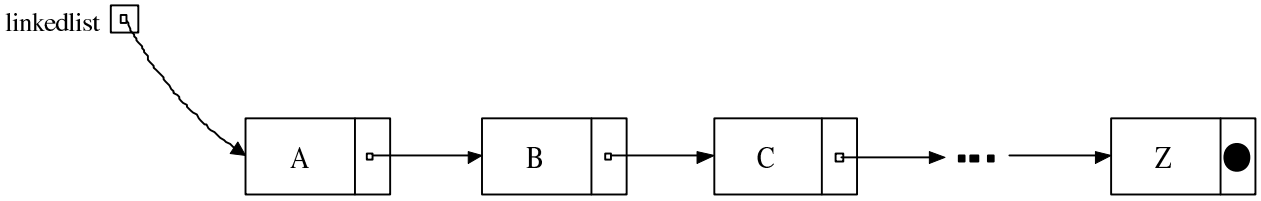
(a) [2 marks] Which collection type keeps items in order, but only allows you to add elements at one end and remove them from the other end?

(b) [2 marks] Suppose an array contains n items (in order). What is the **worst** case asymptotic ("Big O") cost of the binary search algorithm?

O( )

(c) [2 marks] Name a sorting algorithm with a **worst** case asymptotic ("Big O") cost of $O(n \log(n))$.

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



(d) [2 marks] What is the **worst** case asymptotic ("Big O") cost of removing the last item from the above linked list?

O( )

(e) [2 marks] What is the value of linkedlist.next.next.item in the above linked list?

(Question 1 continued on next page)

(Question 1 continued)

(f) [2 marks] State the defining features of Set, Bag, and Map.

(g) [2 marks] Can we have duplicates in a Binary Search Tree?

(h) [2 marks] If we are adding a value to a Hash Table using buckets, and the first bucket we look at already contains a value, where do we put the value?

(i) [2 marks] Draw a *directed* graph containing five nodes and five edges.

Question 2. Sorting [15 marks]

The code on this page shows two sorting algorithms discussed in the lectures, Insertion Sort and Quick Sort. You may find it useful to refer to these functions as you answer the questions in this section. PrintArray is a method that prints out the contents of the array. The questions that follow will make reference to this print statement.

The following is an implementation of Insertion Sort:

```
1  public void insertionSort(E[] data, int size, Comparator<E> comp){
2     for (int i=1; i<size; i++) {
3        E item = data[i];
4        int place = i;
5        while (place > 0 && comp.compare(item, data[place-1]) < 0) {
6           data[place] = data[place-1];
7           place--;
8        }
9        data[place] = item;
10       System.out.print(i + "::    ");
11       PrintArray(data);     // <------ The print statement
12    }
13 }
```

The following is an implementation of Recursive Quick Sort. There are many correct implementations of partition, two were discussed in the lectures:

```
1  public void quickSort(E[] data, int low, int high, Comparator<E> comp) {
2      if(high-low<2)
3          return;      // base case
4      else {
5          int mid = partition(data, low, high, comp);
6          PrintArray(data);     // <------ The print statement
7          quickSort(data,low,mid-1,comp);
8          quickSort(data,mid,high, comp);
9      }
10 }
```

(Question 2 continued on next page)

(Question 2 continued)

(a) [8 marks] Suppose an array containing the following values is to be sorted using Insertion Sort. Show the state of the array printed by the first 5 calls to PrintArray. Refer to the code at the start of the sorting question to find out exactly when PrintArray is called.

Matlab	C++	Algol	Pascal	Haskell	Ruby	Smalltalk	Fortran	Java
0	1	2	3	4	5	6	7	8

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

(Question 2 continued on next page)

(Question 2 continued)

(b) [12 marks] Suppose an array containing the following values is to be sorted.

Matlab	C++	Algol	Pascal	Haskell	Ruby	Smalltalk	Fortran	Java
0	1	2	3	4	5	6	7	8

Your co-worker has tried to sort the array by implementing recursive Quick Sort, but there are problems and the array was not sorted correctly. As the array does not change in the base case, your co-worker has given you an output of the contents of the array after each call to partition. Each step shows the values used for low, high, and pivot in that step. They would like you to help them debug their algorithm. For each step, if there are any errors in the array, circle the errors and write an explanation of why they indicate errors. You should ignore errors that resulted from earlier steps, when highlighting the errors in later steps. To make it easier, your co-worker has also labelled each item with a number indicating its correct position in the array. Note: the code shown at the start of the sorting question section is a correct implementation of recursive quick sort, it is not your co-workers incorrect implementation.

4 Java	0 Algol	1 C++	2 Fortran	3 Haskell	7 Ruby	8 Smalltalk	6 Pascal	5 Matlab	
0	1	2	3	4	5	6	7	8	9
low				pivot					high

Explanation:

0 Algol	1 C++	2 Fortran	4 Java	3 Haskell	7 Ruby	8 Smalltalk	6 Pascal	5 Matlab	
0	1	2	3	4	5	6	7	8	9
low		pivot		high					

Explanation:

0 Algol	1 C++	2 Fortran	4 Java	3 Haskell	5 Matlab	6 Pascal	8 Smalltalk	7 Ruby
0	1	2	3	4	5	6	7	8	9
					low	pivot			high

Explanation:

0 Algol	1 C++	2 Fortran	4 Java	3 Haskell	5 Matlab	6 Pascal	7 Ruby	8 Smalltalk
0	1	2	3	4	5	6	7	8	9
low	pivot	high							

Explanation:

0 Algol	1 C++	2 Fortran	4 Java	3 Haskell	5 Matlab	6 Pascal	7 Ruby	8 Smalltalk
0	1	2	3	4	5	6	7	8	9
							low	pivot	high

Explanation:

Student ID: .....

Question 3. Analysis of Algorithms [15 marks]

For each of the following code fragments, assume that data is an array containing n int values (i.e., data.length is n).

(a) [4 marks] How many lines of data values will be printed out and what is the asymptotic ("Big O") cost expressed in terms of n?

```
1 for(int i = 0; i < data.length; i++)
2     for(int j = 0; j < data.length; j++)
3         System.out.println(data[j]);
```

Number of lines:

"Big O" Cost: O( )

(b) [5 marks] How many lines of data values will be printed out and what is the asymptotic ("Big O") cost expressed in terms of n?

```
1 for(int i = 0; i < data.length; i++)
2     for(int j = 0; j < i; j++)
3         System.out.println(data[j]);
```

Number of lines:

"Big O" Cost: O( )

Student ID: .....

(c) [6 marks] How many lines of data values will be printed out and what is the asymptotic ("Big O") cost expressed in terms of n?

```
1 for(int i = 0; i < data.length; i++)
2     for(int j = i+1; j > i; j--)
3         for(int k = data.length; k > j; k--)
4             System.out.println(data[j]);
```

Number of lines:

"Big O" Cost: O( )

Student ID: .....

Question 4. Programming with Collections [15 marks]

(a) [5 marks] printFile is a method that reads a file, one line at a time, and then writes it out to the screen. You are to complete the method printReverseFile, which reads a file, one line at a time, and then prints it out in reverse order. Your code must use the Java Stack Collection to reverse the order of the items.

```
1  public void printFile(File file) {
2      try {
3          Scanner scanner = new Scanner(file);
4          while(scanner.hasNextLine()) {
5              String line = scanner.nextLine();
6              System.out.println(line);
7          }
8      }
9      catch(FileNotFoundException e){}
10 }
```

```
public void printReverseFile(File file) {









}
```

(b) [10 marks] A small company has one video camera and they would like to make a simple system for keeping track of reservations for use of the camera. There is at most one reservation of the camera for any one date. The reservation system needs to have three methods. You are to complete the code below to implement these three methods and to define the collection to hold the reservations.

The first method is makeReservation, which takes a date and name, and either makes the booking and prints out that the booking has been made or it says that the date has already been booked.

The second method is checkReservation, which takes a date, and either prints out that there is no booking for that date, or prints out who has reserved the camera on that date.

The third method is bookingSummary, which prints out a summary of who has bookings in the system. It should print out the names of all the people with bookings in the system, and along with each name should be the number of bookings that person has in the system.

Student ID: .....

```
public class ReservationSystem {

  private                    reservations = new

  public void makeReservation(Date date, String name) {









  }

  public void checkReservation(Date date) {









  }

  public void bookingSummary() {

















  }
}
```

Question 5. Binary Search Trees [25 marks]

(a) [3 marks] Draw a Binary Search Tree containing the letters G, A, Z, X, P, N, and E. The root should contain the letter N.

(b) [2 marks] State the difference between Binary Tree and a Binary Search Tree?

(c) [3 marks] Describe the difference between iterative and recursive algorithm?

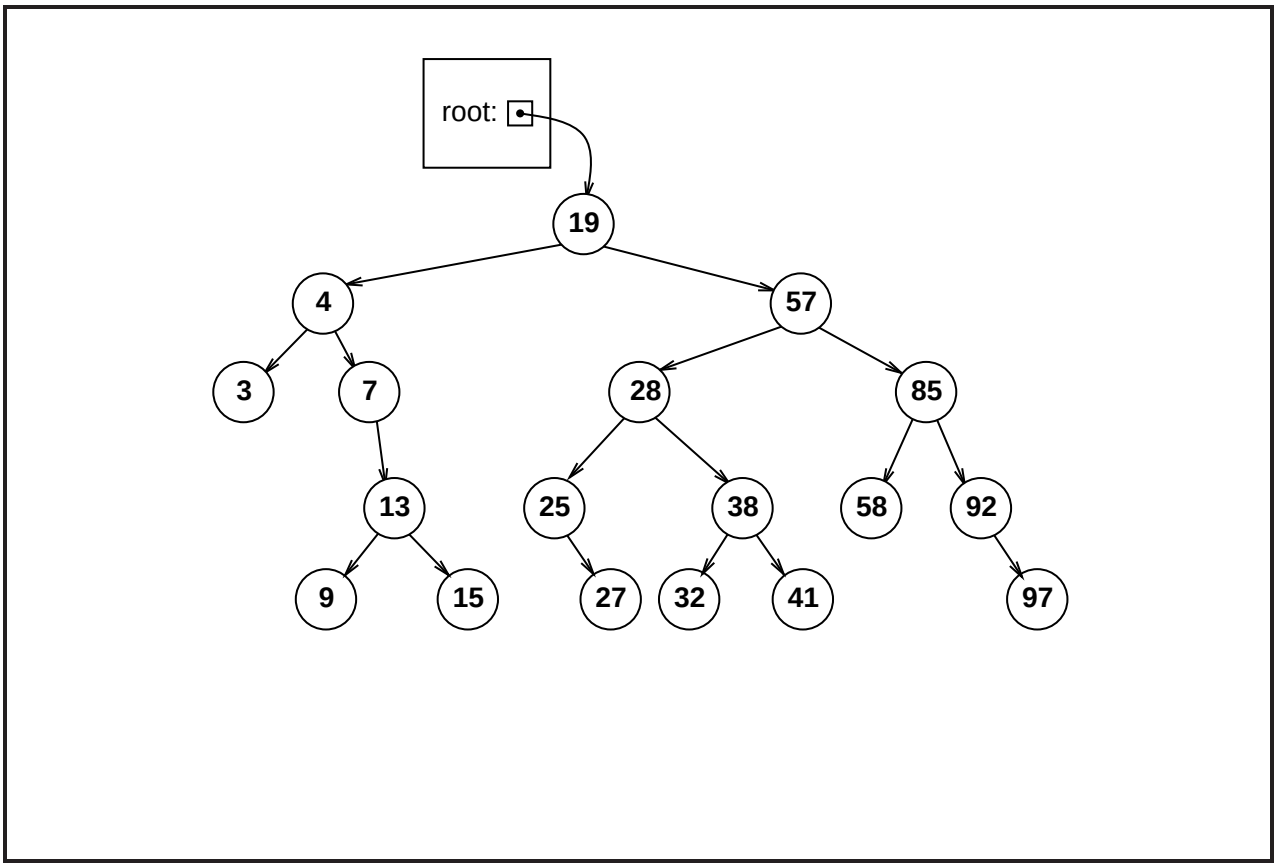
(d) [2 marks] Consider removal of a node from a Binary Search Tree when the node has two children. What will happen if during the removal we replace the node with two children with rightmost in the left sub tree instead of leftmost in the right sub tree?

(Question 5 continued on next page)

(Question 5 continued)

(e) [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?

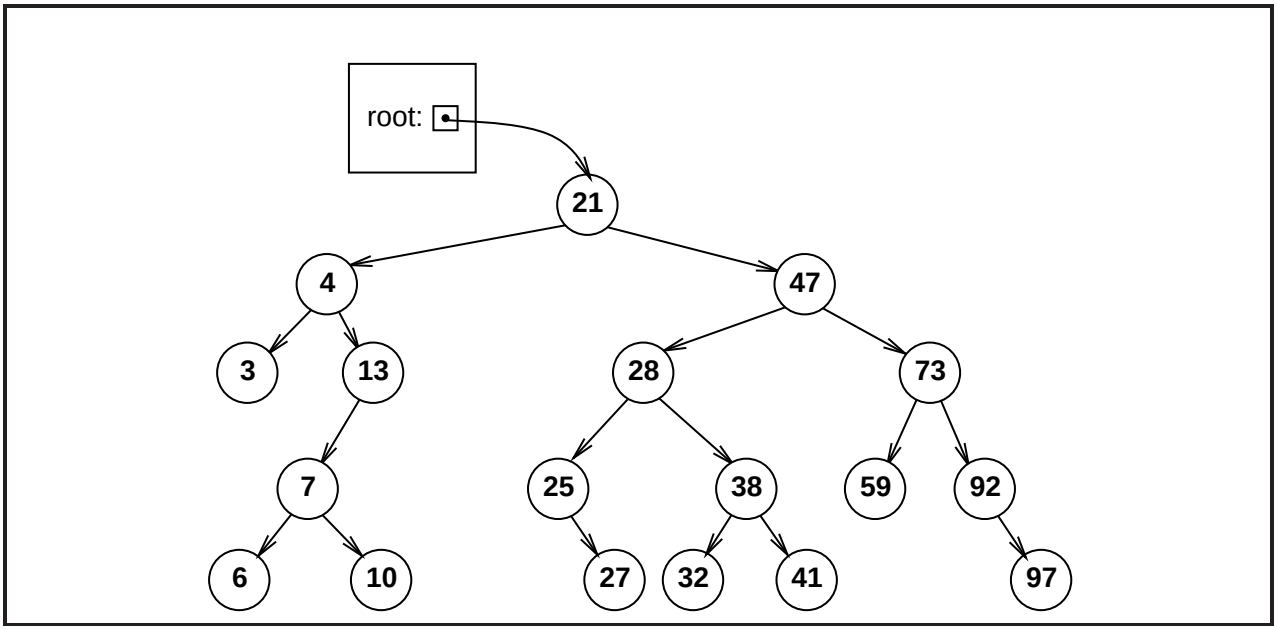
(f) [4 marks] Consider the following diagram of a Set of numbers implemented as a Binary Search Tree. Show, on the diagram, what the tree would look like if the values 8, 26, 8, and 84 were added to the set.



(Question 5 continued on next page)

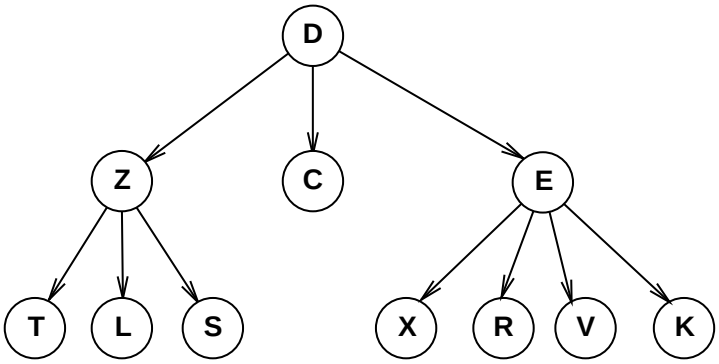
(Question 5 continued)

(g) [8 marks] In the blank space below, draw what the following binary search tree would look like if the values 6, 28, 21, and 47 were removed, in that order. Either use the removal algorithm presented in class or state clearly the algorithm you are using.



Question 6. General Trees [18 marks]

(a) [4 marks] Write out the order in which nodes would be visited by a depth-first, right-to-left, in-order traversal of the following general tree.



(b) [2 marks] Iterative depth-first traversal uses explicit stack. It is missing from a recursive traversal. Where is the stack when doing depth-first recursively?

(Question 6 continued on next page)

(Question 6 continued)

(c) [5 marks] Given a GeneralTree implementation (this is the same as GeneralTree used in assignment 7), complete the following method that returns the size of the tree.

```
public class GeneralTree<KeyType> {
    private KeyType _rootValue;
    private Set<GeneralTree<KeyType>> _children;


    public int size() {









    }
}
```

(d) [2 marks] Why does an in-order traversal of a ternary tree make no sense?

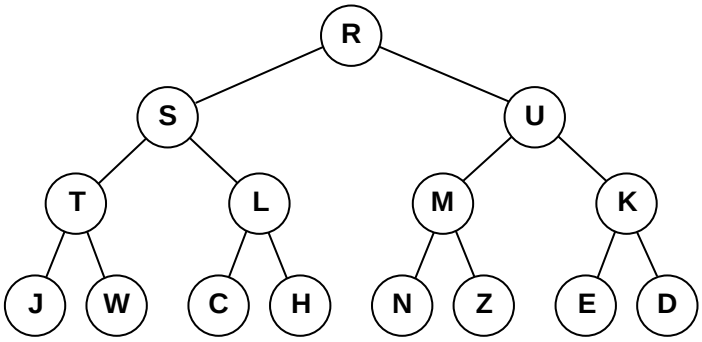
(Question 6 continued on next page)

(Question 6 continued)

(e) [5 marks] The following traversal method below does a depth first traversal of a binary tree using a stack.

```
1  public static void traversal(TreeNode start) {
2      Stack<TreeNode> toVisit = new Stack<TreeNode>();
3      toVisit.push(start);
4      while (!toVisit.isEmpty()) {
5          TreeNode current = toVisit.pop();
6          System.out.println(current.value);
7          if (current.getLeft() !=null)
8              toVisit.push(current.getLeft());
9          if (current.getRight() != null)
10             toVisit.push(current.getRight());
11     }
12 }
```

Suppose traversal is called on the following tree:



Draw the contents of the stack immediately after node "N" has been added to the stack.

Stack:

Question 7. Heaps [23 marks]

(a) [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: X (3), B (2), C (7), T (4)

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

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

Data Items: N (2), E (5), L (2), S (3)

The HeapQueue class is an implementation of priority queues using a heap — a complete, partially ordered binary tree in an array.

(c) [2 marks] If a complete partially ordered binary tree is stored in an array, and the index of the root is 0, what are the indices of the children of the node at index *i* and the index of the parent of the node at index *i*?

(Question 7 continued on next page)

(Question 7 continued)

(d) [1 mark] If a complete partially ordered binary tree is stored in an array, and the index of the root is 0, what is the index of the parent of the node at index *i*?

(e) [6 marks] Complete the following pushup(*int* i) method that moves the value at index i up the partially ordered tree into its correct position.
Assume that the HeapQueue contains the following fields:

```
private E[ ] data;
private int count;
private Comparator<E> comp;
```

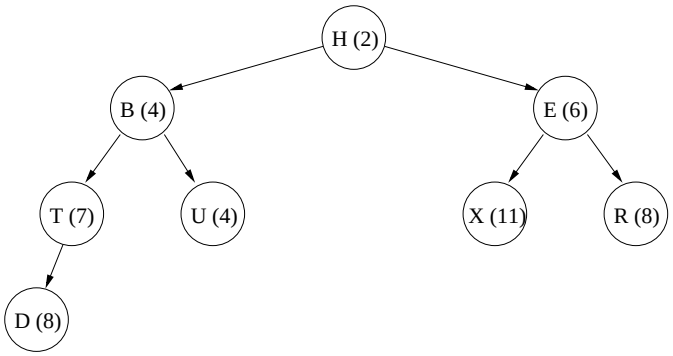
where comp is a comparator that considers values with higher priority to be larger than values with lower priority.

```
private void pushup(int i){
    if (i == 0) return;




}
```

(Question 7 continued on next page)

(Question 7 continued)

Suppose a heap contains the following items:



(f) [4 marks] Draw the new heap after *dequeue()* has been called twice on the heap above.

(g) [3 marks] Draw the heap after *dequeue()* has been called another two times on the heap resulting from the previous question.

Question 8. Priority Queues [18 marks]

This question concerns a ManagePatients program that helps a hospital keep track of the patients it has to treat in the waiting room. The hospital processes patients in order of their priority, so the ManagePatients program uses a priority queue.

The program has three actions:

New Patient Asks the user for the details of a new patient, adds it to the priority queue, and prints out its details.

Next Patient Removes the patient at the front of the priority queue and prints out the details of the patient, or a useful message if there are no more patients to treat.

Report Queue Prints the number of patients in the priority queue, and all their details.

The ManagePatients class on the facing page contains an incomplete method for each of these actions.

The ManagePatients class uses the Patient class described below, and the standard Queue interface described in the appendix.

```
public class Patient implements Comparable<Patient>{
    private String category;          //One of "critical", "urgent", or "non urgent"
    private int timeOfArrival;        //time of arrival used to calculate waiting time
        ⋮

    /** Constructor asks the user for the details of a new patient
        and fills in the field values */
    public Patient(String prompt){
            ⋮
    }
    /** Returns a String with all the details of the patient */
    public String getDetails(){
            ⋮
    }

        ⋮
}
```

(Question 8 continued on next page)

(Question 8 continued)

(a) [12 marks] Complete the three methods in the ManagePatients class below.

```
public class ManagePatients {

    // Priority queue of all patients
    private Queue <Patient> patients= new HeapQueue<Patient>();

    public ManagePatients(){
        ⋮
    }

    /** Get new patient details from user, add it to the queue,
        print details to System.out */
    public void enqueueNewPatient(){









    }

    /** Take patient off front of queue and print its details.
        If queue is empty, print "No Patients" */
    public void NextPatientToProcess(){









    }

    /** Print how many patients are waiting, and all their details */
    public void reportQueue(){









    }
}
```

(Question 8 continued on next page)

(Question 8 continued)

There are three categories of patients: "critical", "urgent", and "non urgent". All the "critical" patients are higher priority than the "urgent" packages, which are higher priority than the "non urgent" packages. Within each category, the patients that have waited the longest have the highest priority.

(b) [6 marks] Two efficient implementations of a priority queue are:

- an array of Queues, indexed by the priority, one queue for each priority value, and
- a HeapQueue, using a partially ordered tree in an array.

Given the above definition of priority, which implementation would be a good choice for the Manage Patients program? Explain why.

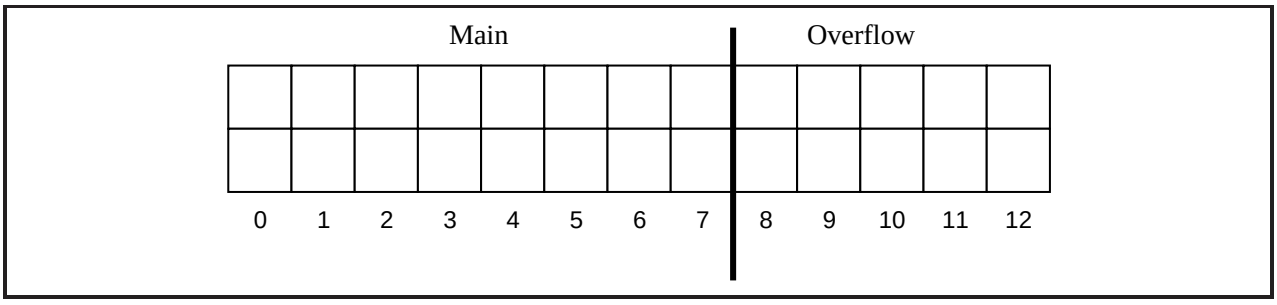
Question 9. Hashing [18 marks]

(a) [3 marks] Typically we increase the capacity of a Hash Table with probing when it is only 70% full. Why don't we wait until the Hash Table is 100% full?

(b) [2 marks] VUW Student ID's consist of 8 digits. Most recent ID's start with the digits "3000". Suggest a good hash function for VUW student ID's.

(c) [5 marks] Draw the contents of the array after the following 10 values are inserted into a Hash Table that uses an overflow area to handle collisions.
Note: do *not* increase the size of the array if the table becomes too full.

value:	'B'	'A'	'J'	'K'	'E'	'S'	'L'	'M'	'Z'	'Y'
hashed index:	1	6	3	0	6	5	1	6	4	1



(Question 9 continued on next page)

(Question 9 continued)

The OpenHashBag class implements the *Bag* interface using a hash table with probing. Part of its code is given below.

```
public class OpenHashBag implements Bag {
    private Object[ ] data;
    private int count = 0;

    public OpenHashBag (int size) {
        data = new Object[size];
    }

            ⋮

    private int hash(Object val){
        if (val = = null) throw new NoSuchElementException();
        return Math.abs(val.hashCode()) % data.length;
    }
    ⋮
}
```

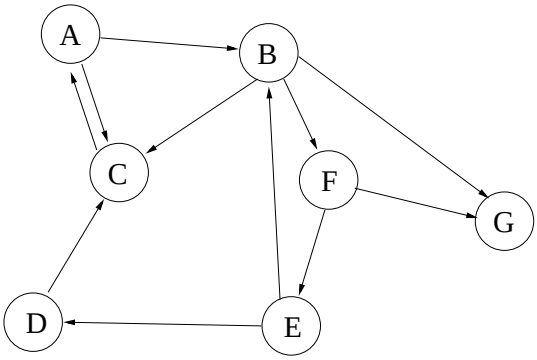
(d) [8 marks] Complete the following code for the containsElement method, using linear probing. containsElement returns true if an element of the Bag matches the argument. It returns false if there is no matching element in the bag.

```
public boolean containsElement (Object val) {









}
```

Question 10. Graphs

[10 marks]

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



(Question 10 continued on next page)

(Question 10 continued)

(b) [2 marks] State one reason why a tree traversal algorithm might not work on a directed graph.

(c) [2 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?

****

Appendices

Possibly useful formulas:

- $1 + 2 + 3 + 4 + \cdots + k \;=\; \frac{k(k+1)}{2}$
- $1 + 2 + 4 + 8 + \cdots + 2^k \;=\; 2^{k+1} - 1$
- $a + (a+b) + (a+2b) + \cdots + (a+kb) \;=\; \frac{(2a+kb)(k+1)}{2}$
- $a + as + as^2 + as^3 + \cdots + as^k \;=\; \frac{as^{k+1}-a}{s-1}$

Table of base 2 logarithms:

n	1	2	4	8	16	32	64	128	256	512	1024	1,048,576
$\log(n)$	0	1	2	3	4	5	6	7	8	9	10	20

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

```
public class Scanner {
    public boolean hasNext(); //there is more to read
    public String next(); //return the next token (word)
    public String nextLine(); //return the next line
    public int nextInt(); //return the next integer
}
public interface Iterator<E> {
    public boolean hasNext();
    public E next();
    public void remove();
}
public interface Comparable <E>{
    public int compareTo(E o);
}
public interface Comparator <E>{
    public int compare(E o1, E o2);
}
public class Math{
    public static double random(); //return a random number between 0.0 and 1.0
}
public interface Collection <E>{
    public boolean isEmpty ();
    public int size ();
    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 (E element);
    public void add (int index, E element);
    public void remove (int index);
    public void remove (Object element);
}
public interface Set extends Collection <E> {
    // Implementations: ArraySet, SortedArraySet, HashSet
    public boolean contains (E element);
    public void add (E element);
    public void remove (Object element);
}
public interface Map <K, V> {
    // Implementations: HashMap, TreeMap, ArrayMap
    public V get (K key); //returns null if no such key
    public void put (K key, V value);
    public void remove (K key);
    public Collection<V> values ();
    public Set<Map.Entry<K, V> > entrySet ();
}
public interface Map.Entry <K, V> {
    public K getKey ();
    public V getValue ();
}
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 ();
    public E pop ();
    public E push (E element);
    public boolean empty ();
}
public class Arrays {
    public static <E> void sort(E[ ] ar, Comparator<E> comp);
}
public class Collections {
    public static <E> void sort(List<E> list, Comparator<E> comp);
}
```

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.