

EXAMINATIONS — 2010

END YEAR

COMP103
Introduction to
Data Structures and Algorithms
SOLUTIONS

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 and interfaces can be found at the end of the paper.

Questions	Marks
1. Basic Questions	[26]
2. Using Collections	[25]
3. Implementing Collections	[23]
4. Recursion, and Sorting	[20]
5. Linked Lists, and Trees	[28]
6. Binary Search Trees	[17]
7. Partially Ordered Trees and Heaps	[22]
8. Various Topics	[19]

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

[26 marks]

(a) [2 marks] State one difference between an Array and an ArrayList.

ArrayList grows automatically, and has get/set/size etc methods.

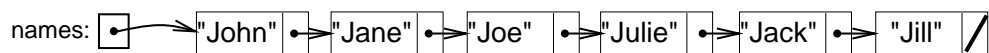
(b) [2 marks] What is the average case Big-O cost of searching for an item in a Set of n items implemented using an unsorted array?

$O(n)$

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

Insertion sort, or Selection sort, or Bubble sort

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 value of names.next.item?

"Jane"

(e) [2 marks] What is the value of names.next.next.next.item?

"Julie"

(Question 1 continued on next page)

(Question 1 continued)

(f) [10 marks] Draw lines between the physical things on the left and the collections on the right indicating the best choice of collection for that physical thing. Note: it may or may not be a 1:1 mapping.

Clothes in a load of washing.	Set	QUEUE
Layers in sedimentary rock.	Stack	SET
Music notes in a tune.	List	STACK
Cars in a drive-through.	Queue	MAP
Museums and their phone numbers.	Map	LIST

(g) [2 marks] What is the maximum number of leaves in a binary tree of depth 3 (four levels of nodes)?

8

(h) [2 marks] If a Bag is implemented using an array of values, would the *addElement* method be faster if the values were sorted or un-sorted?

The addElement method would be faster for an un-sorted collection

(i) [2 marks] What is the name of a good algorithm for finding an item in a sorted array of items.

Binary Search

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. Using Collections

[25 marks]

This question requires you to complete three methods for a program for processing enrolment data for a university. The program reads a couple of files, and then prints out a list of schools for each student, and a list of students for each school.

(a) [6 marks] The first method reads the `courses-schools.txt` file that lists all the courses in the university and the schools that teach them, and constructs a **Map** for looking up the school that teaches a given course. The file has one line for each course, containing the course code and the abbreviation of the school. For example:

```
COMP102    ECS
COMP103    ECS
MATH114    SMSOR
PHYS105    SCPS
TECH102    SCPS
BIOL123    SBS
...
```

Complete the `readCourses()` method below that will initialise and fill the `courseToSchool` field. Documentation for `Set`, `Map`, `List`, etc. is given in an appendix at the end of this exam.

```
import java.util.*;
public class Enrolment{

    private Map<String, String> courseToSchool; // map from course name to school name

    /* Reads a file of school – course pairs into a map indexed by course */
    public void readCourses(){

        courseToSchool = new HashMap<String, String>();

        try {
            Scanner sc = new Scanner(new File("courses-schools.txt"));
            while ( sc.hasNext() ){

                String course = sc.next();
                String school = sc.next();
                courseToSchool.put(course, school);

            }
            sc.close();
        }
        catch(Exception e){System.out.println("readCourses failed");}
    }
}
```

(Question 2 continued on next page)

(Question 2 continued)

Another file (`enrolments.txt`) contains the list of courses that each student has enrolled for. Each line of this file starts with a student ID followed by the course codes. An example file might be:

```
4000675123  COMP102 COMP103 MATH114 TECH102 BIOL103
4001725125  BIOL101 BIOL102 BIOL103
4001997126  COMP102 BIOL101 COMP103 BIOL103
...
```

(b) [9 marks] The second method should print out the schools that each student needs approval from. *It should not list a school more than once per row, even if the student is taking several courses from the school.* For example, given the three students above, it should print out this:

```
4000675123:  ECS SMSOR SCPS BIO
4001725125:  SBS
4001997126:  ECS SBS
```

Complete the `listsOfSchools()` method below. For each student, the method will need to read each course and look up the school in the `courseToSchool` map. It will need a collection to keep track of the schools for the current student. *Nb:* remember that all three methods are in the same class.

```
public void listsOfSchools(){
    try {
        Scanner sc = new Scanner(new File("enrolments.txt"));
        while ( sc.hasNext() ){
            int ID = sc.nextInt ();           // read student ID

            Set<String> goSee = new HashSet<String>();

            while ( sc.hasNext() && !sc.hasNextInt() ){ // read courses
                String course = sc.next ();
                String school = courseToSchool.get(course);
                goSee.add(school); // add automatically rejects duplicates
            }
            System.out.print(ID+": "); // print out ID and list of schools
            for (String school : goSee)
                System.out.print(school + " ");
            System.out.println ();

        }
        sc.close ();
    } catch (Exception e){System.out.println("listsOfSchools failed");}
}
```

(Question 2 continued on next page)

(Question 2 continued)

(c) [10 marks] The third method should print out a list for each school containing all the students taking courses in the school. *Nb*: remember that all three methods are in the same class.

Complete the `listsOfStudents()` method below. The method will need to read the whole of the `enrolments.txt` file, building a data structure that keeps track of all the students taking courses in each school. It will then print out the list of students for each school.

```
public void listsOfStudents(){
    try {
        // Map with key being a school
        Map<String, Set<Integer>> schoolToIDs = new HashMap<String, Set<Integer>>();
        Scanner sc = new Scanner(new File("enrolments.txt"));
        while ( sc.hasNext() ){
            int ID = sc.nextInt ();
            while ( sc.hasNext() && !sc.hasNextInt() ){
                String course = sc.next ();
                String school = courseToSchool.get(course);
                Set<Integer> students = schoolToIDs.get(school);
                if ( students == null ){
                    students = new HashSet<Integer>();
                    schoolToIDs.put(school, students);
                }
                students.add(ID); // no need to put the set back – it's already there
            }
        }
        sc.close ();
        // Print out lists
        for (String school : schoolToIDs.keySet()){
            System.out.print(school + " : ");
            Set<Integer> students = schoolToIDs.get(school);
            for ( int ID : students)
                System.out.print(ID + " ");
            System.out.println ();
        }
        or
        for (Map.Entry<String, Set<Integer>> entry : schoolToIDs.entrySet()){
            System.out.print(entry.key() + " : ");
            for ( int ID : entry.value())
                System.out.print(ID + " ");
            System.out.println ();
        }
    }
    catch (Exception e){System.out.println("listsOfStudents failed");}
}
```


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 3. Implementing Collections

[23 marks]

A Set is a type of collection that has no structure, but it is not allowed to contain duplicates. Part of the code for the `ArraySet` class is shown below.

```
import java. util .*;

public class ArraySet<E> extends AbstractCollection<E> {
    private static int INITIALCAPACITY = 10;
    private int count = 0;
    private E[] data;

    public ArraySet() {
        data = (E[]) new Object[INITIALCAPACITY];
    }

    public boolean add(E item) {
        ...
    }

    public boolean contains(Object item) {
        ...
    }

    public boolean remove(Object item) {
        if (item == null) return false;
        for (int i = 0; i < count; i++) {
            if (data[i].equals(item)) {
                count--;
                data[i] = data[count];
                return true;
            }
        }
        return false;
    }

    private void ensureCapacity () {
        if (count < data.length) return;
        E [ ] newArray = (E [ ])(new Object[data.length*2]);
        for (int i = 0; i < count; i++) newArray[i] = data[i];
        data = newArray;
    }

    public Iterator <E> iterator() {
        return new ArraySetIterator <E> (this);
    }
}
```

(Question 3 continued on next page)

(Question 3 continued)

You are to complete two methods of the `ArraySet` class below. Before doing so, note that:

- `ArraySet` does not need to keep items in order.
- this implementation of `ArraySet` does not use a separate “`findIndex`” method.

(a) [5 marks] Complete the `contains(Object item)` method, that will return `true` if the `ArraySet` contains a value equal to `item` and will otherwise return `false`.

```
public boolean contains(Object item) {
    if (item==null) return false;
    for (int i=0; i<count; i++)
        if (item.equals(data[i]))
            return true;
    return false;
}
```

(b) [6 marks] Complete the `add` method which will insert an item into the set, unless the item is already present. `add` will return `true` if and only if it modifies the set. Before adding an item, `add` should call the `ensureCapacity` method to make sure there is sufficient room in the `data` array.

```
public boolean add(E item){
    for (int i=0; i<count; i++){
        if (item.equals(data[i]))
            return false;
    }
    ensureCapacity();
    data[count]=item;
    count++;
    return true;
}
```

(Question 3 continued on next page)

(Question 3 continued)

(c) [8 marks] Complete the following `ArraySetIterator` class that defines an iterator for an `ArraySet`. Note that `ArraySetIterator` is a private inner class of `ArraySet`.

The constructor, and `remove`, have been done for you.

```
private class ArraySetIterator <E> implements Iterator <E> {
    private ArraySet<E> set;

    private ArraySetIterator (ArraySet<E> s) {
        set = s;
    }

    public boolean hasNext() {

    }

    public E next() {

    }

    public void remove() {
        throw new UnsupportedOperationException();
    }
}
```

(Question 3 continued on next page)

(Question 3 continued)

(d) [4 marks] Write a ReverseRobotComparator that compares two Robots (i.e. 2 items of type Robot) in reverse (decreasing) order according to their battery life. Assume that the class Robot has a method getBatts(), which returns the battery life of a Robot as an integer.

```
private static class ReverseRobotComparator implements ... Comparable<Robot> {  
  
    public int compare(Robot r1, Robot r2) {  
        return (r1.getBatts() - r2.getBatts ());  
    }  
  
}
```

Question 4. Recursion and Sorting

[20 marks]

Consider the following code:

```
public int recursiveFunc(int n) {  
    if (n==0)  
        return(1);  
    else  
        return( recursiveFunc(n-1) + 1 );  
}
```

(a) [3 marks] What value is returned by recursiveFunc(150)?

151

(b) [2 marks] What is the time complexity ("big- \mathcal{O} " cost) of the method recursiveFunc, expressed in terms of the parameter n ?

$O(n)$

Consider another recursive method:

```
public int recursiveFunc2(int n) {  
    if (n == 0)  
        return(0);  
    else  
        return( recursiveFunc2(n/2)+1 );  
}
```

(c) [3 marks] What value will be returned by calling recursiveFunc2(17) ?

5

(d) [2 marks] What is the time complexity ("big- \mathcal{O} " cost) of the method recursiveFunc2, expressed in terms of the parameter n ?

$O(\log_2 n)$

(Question 4 continued on next page)

(Question 4 continued)

(e) [2 marks] List one advantage and one disadvantage, of Merge Sort over Quick Sort.

Advantage: MergeSort is stable, QuickSort is not.

Disadvantage: QuickSort is in-place (no temporary data), while MergeSort is not.

Consider the following code:

```
public int split ( int low, int high) {
    if (low+1 >= high) return 1;
    int mid = (low+high)/2;
    int lowResult = split (low,mid);
    int highResult = split (mid,high);
    return(lowResult+highResult+1);
}

public int mergeVal(String s) {
    return split (0,s.length ());
}
```

(f) [4 marks] What value would be printed to the screen if `mergeVal` is called in the following code? (*Hint: consider drawing the binary tree of recursive calls*)

```
System.out.println(mergeVal("abcdefgh"));
```

15

(g) [4 marks] If the length of the parameter string, s , is n , what is the big- \mathcal{O} cost of `mergeVal`?

$O(n)$

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.

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. Linked Lists and Trees

[28 marks]

(a) [4 marks] Write an iterative or recursive method, `lastActionHero`, that returns the last value (of type `ActionHero`) from a list of `ActionHeros` implemented as a list of `LinkedLists`.

```
/**Returns the value in the last node of a list starting at a node */
public ActionHero lastActionHero (LinkedList<ActionHero> list){

    if ( list == null)        return null;
    LinkedList<ActionHero> rest = list;
    while(rest.next!=null)
        rest = rest.next ();
    return rest.get ();

}
```

(b) [4 marks] In a "ternary" tree, each node has up to 3 children. If a ternary tree is complete, and contains 40 nodes, what is its height?

Height is 3: It has 1 at the root, 3 at level 1, 9 at 2 and 27 at 3. $1+3+9+27=40$

(c) [10 marks] Consider the following `TreeNode` class:

```
public class TreeNode {
    private String word;
    private Set<TreeNode> children = new HashSet<TreeNode>();
    :
    :
}
```

In the box below, complete the `find` method in a `TreeNode` class. This should use recursion to search the tree below `node` for a node having a field `word` matching the argument `text`.

```
private TreeNode find(TreeNode node, String text) {
    if (node.word.equals(text))
        return node;
    for (TreeNode ch: node.children) {
        TreeNode ans = find(ch, text);
        if (ans != null) return ans;
    }
    return null;
}
```

(d) [6 marks] In lectures, we discussed the use of a `Queue` for carrying out a breadth-first traversal of a general tree. In the box below, give *pseudocode* for this iterative algorithm.

```
make a q,
put root on the q,
while q not empty:
    poll q
    process node
    put its children on the q
```

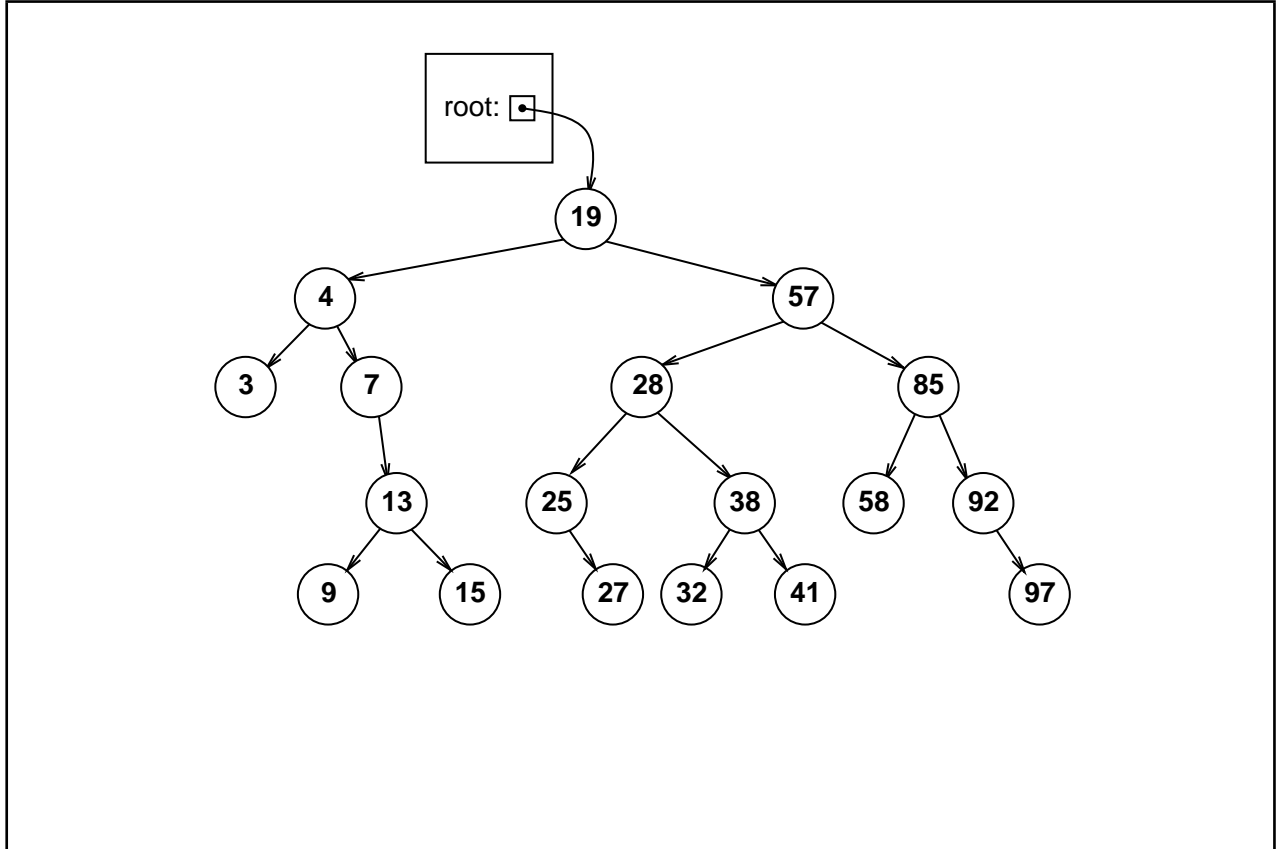
(e) [4 marks] A simple change to the algorithm in (d) would result in a different kind of traversal of the same tree. What is the change, and what kind of traversal does it result in?

A stack instead of a Queue leads to depth-first pre-order traversal.

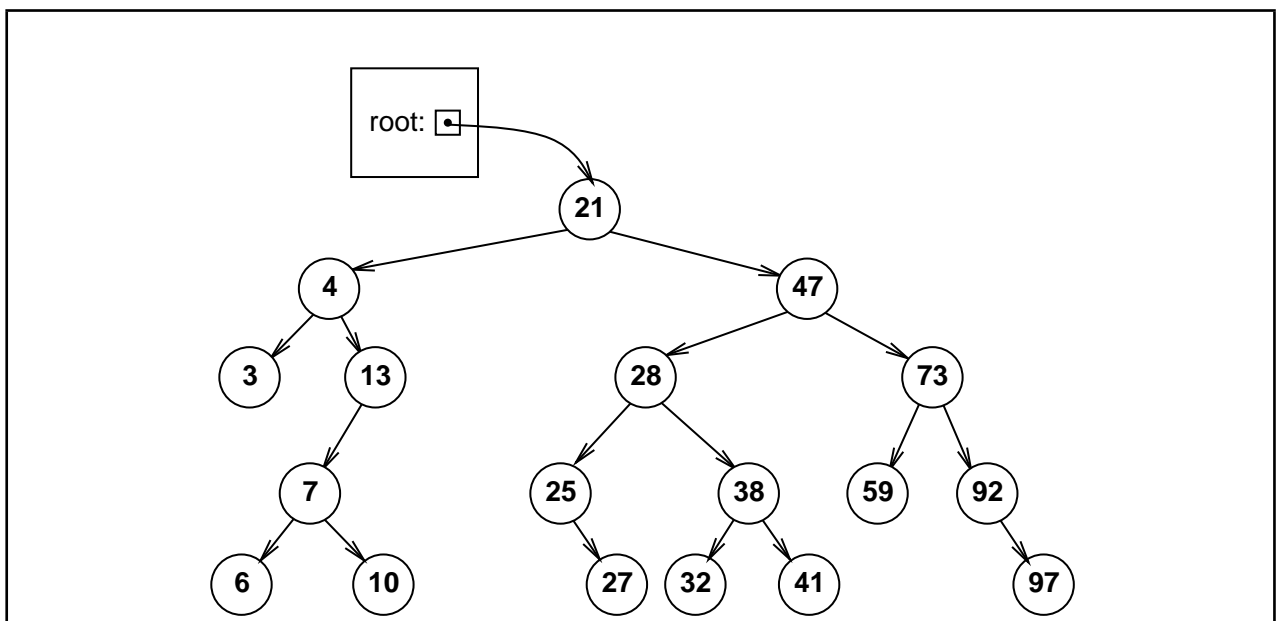
Question 6. Binary Search Trees

[17 marks]

(a) [4 marks] Consider the following diagram of a Set of numbers implemented as a Binary Search Tree. On the diagram, show what the tree would look like if the values 2, 5, 20, and 56 were added to the set.



(b) [6 marks] On the diagram, show what the tree below would look like if the values 32, 7, and 47 were removed from the Set.



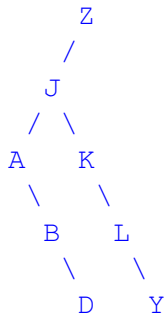
(Question 6 continued on next page)

(Question 6 continued)

(c) [3 marks] Suppose a binary search tree was constructed by inserting elements in a strictly *decreasing* order (ie. with the largest elements coming first). Describe the structure of the tree.

The binary search tree will be entirely unbalanced, with no branching. The structure is equivalent to a linked list. The branching will be to the left of the root.

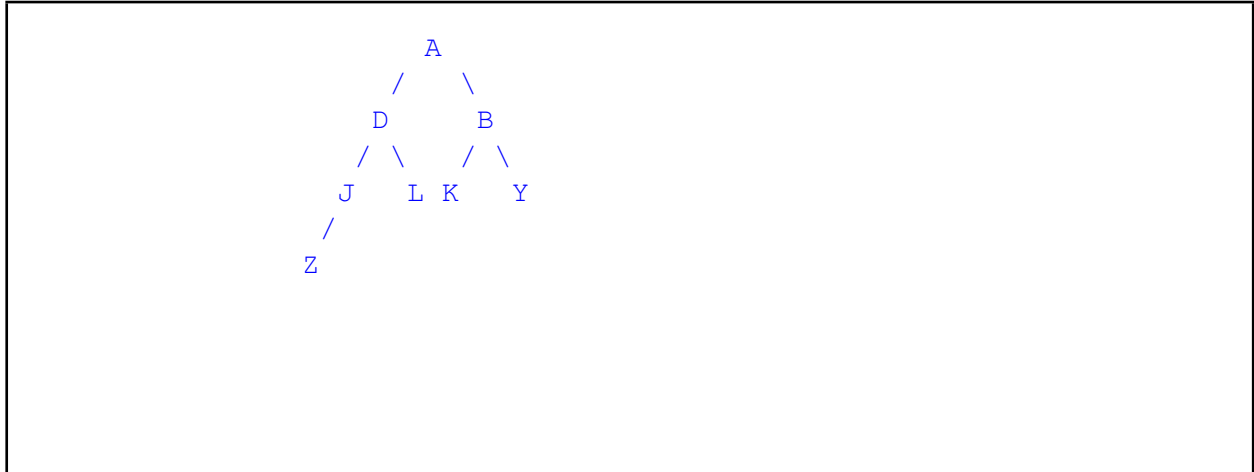
(d) [4 marks] Suppose the items Z, J, K, L, A, B, Y, and D are added, in that order, to a Binary Search Tree that starts off empty. Draw the resulting tree.



Question 7. Partially Ordered Trees and Heaps

[22 marks]

(a) [4 marks] Suppose the items Z, J, K, L, A, B, Y, and D are added, in that order, to a heap that starts off empty. Assuming that letters towards the beginning of the alphabet have higher priority, draw the resulting heap as a tree.



(b) [2 marks] State the property a binary tree must satisfy in order to be a partially ordered tree.

The value in each node must be less than (or equal to) the values in its children.

(c) [2 marks] 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. Suppose a node in a heap has a parent node (i.e. it is not the root node). If this node is at position i in the array which stores the heap, what is the position of its parent node?

Parent is at $(i - 1) / 2$

(d) [2 marks] what are the indices of the children (if they exist)?

Children are at $2i + 1$ and $2i + 2$

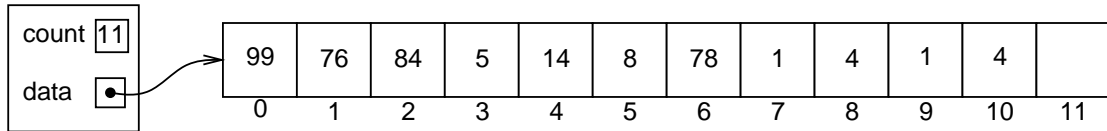
(e) [2 marks] What is the big-O cost of heapify?

$O(n)$

(Question 7 continued on next page)

(Question 7 continued)

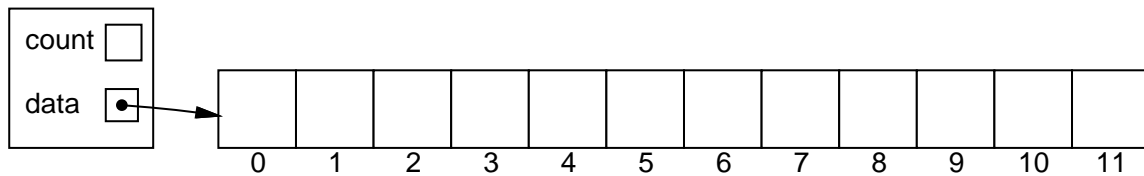
Consider the following **HeapQueue**, where the integer represents the priority of each element in the array (larger numbers are higher priority). The number under each box is just its index in the array.



(f) [5 marks] Show the state of the **HeapQueue** after `poll()` has been called on it once.

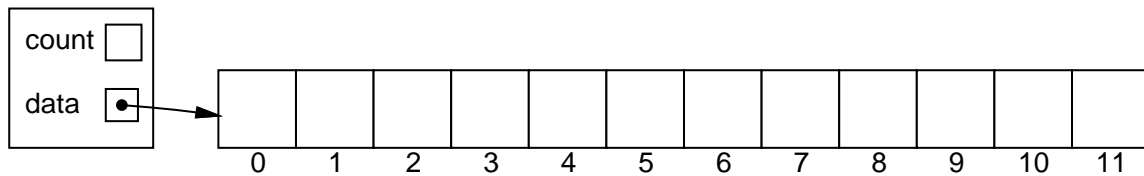
Hint: it might help to first draw the heap as a tree.

After poll:



(g) [5 marks] Re-starting from the **HeapQueue** shown *at the top of the page*, show its state if a new element with priority 21 is offered to the queue.

After offer:



Question 8. Various Topics

[19 marks]

(a) [11 marks] Complete the table below, which gives the asymptotic ("big O") average costs of operations under 3 different implementations of the Bag interface. The first one is done for you.

Data Structure	<i>Adding</i>	<i>Finding</i>	<i>Removing</i>
Unsorted Array	$O(1)$	$O(n)$	$O(n)$
Sorted Array	$O(n)$	$O(\log n)$	$O(n)$
Linked List (unsorted)	$O(1)$	$O(n)$	$O(n)$
Binary Search Tree (assuming it is balanced)	$O(\log n)$	$O(\log n)$	$O(\log n)$

Suppose a hash function for string values returns an integer between 0 and 2^{32} , and a hash table is of size 37.

(b) [2 marks] How could the hash function value be restricted to index the hash table?

The hash function value is reduced modulus the size of the hash table (% 37 in this case)

(c) [2 marks] Why is the value 37 better for the size of the hash table, than 39?

37 is a prime number, whereas 39 is not (it has prime factorisation {13,3})
It is a mathematically proven fact that on randomly sampled data, the hash function will have fewer collisions if the values are reduced to a prime modulus, rather than non-prime modulus.

(d) [4 marks] Give two other attributes the hash function should possess to make it a good hash function.

- easy / fast to compute
- tend to be "evenly" distributed across the array

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.

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.

Appendix (may be removed)

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

```
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 add();
    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(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);
```

```
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 Set<Map.Entry<K, V>> entrySet();
```

```
public class ListNode<E>
    public E get ();
    public void set(E item);
    public ListNode<E> next();
    public void setNext(ListNode<E> nextNode);
```