

## SEMESTER 2 EXAMINATION 2010/2011

## DATA STRUCTURES AND ALGORITHMS

Duration: 120 mins

---

You must enter your Student ID and your ISS login ID (as a cross-check) on this page. You must not write your name anywhere on the paper.

Student ID:	<input type="text"/>	Question	Marks
		1	
		2	
ISS ID:	<input type="text"/>	3	
		4	
		Total	

---

*Answer THREE questions out of FOUR.*

*This examination is worth 85%. The tutorials were worth 15%.*

*University approved calculators MAY be used.*

*Each answer must be completely contained within the box under the corresponding question. No credit will be given for answers presented elsewhere.*

*You are advised to write using a soft pencil so that you may readily correct mistakes with an eraser.*

*You may use a blue book for scratch—it will be discarded without being looked at.*

**Question 1**

(a) Fill in the implementation for a simple resizable array.

*(10 marks)*

```
public class ArrayList<T>
{
    private T[] elements;
    private int noElements;

    public MyArrayList() {

    }

    public T get(int index) throws Exception {

    }

    public void addToEnd(T newElement) {

    }

    public int size() {

    }
};
```

- (b) Assume an ArrayList has an initial capacity of 8 and we add 100 elements (where the capacity is doubled each time it is exceeded) write down the sum of all copy operations that have to be carried out. (3 marks)

---



---



---

- (c) Assuming an initial capacity of  $n$  and we add  $N$  elements write a bound on the number of times the arrays are copied,  $m$ . (4 marks)

---



---



---



---

- (d) Write down an upper bound on the sum of all copy operations that have to be carried out in part (c). Show your working. (8 marks)

---



---



---



---



---

- (e) Give the average (amortised) time complexity for an ArrayList and a doubly linked list for the following operations (8 marks)

	ArrayList	Doubly linked list
Adding an element to end of list		
Adding an element to beginning of list		
Accessing the $i^{th}$ element		
Searching for an element		

End of question 1

Q1: (a)  $\frac{1}{10}$  (b)  $\frac{1}{3}$  (c)  $\frac{1}{4}$  (d)  $\frac{1}{8}$  (e)  $\frac{1}{8}$  Total  $\frac{1}{33}$

**TURN OVER**

(a) The figure below shows a curve of a sort algorithm with time complexity plotted on a log-log axes

[illegible]

The quicksort algorithm is as follows

```
QUICKSORT(a, left, right) {  
    if (right-left < threshold)  
        INSERTIONSORT(a, left, right)  
    else  
        pivot = CHOOSEPIVOT(a, left, right)  
        part = PARTITION(a, pivot, left, right)  
        QUICKSORT(a, left, part)  
        QUICKSORT(a, part+1, right)  
    endif  
}
```

- (b) Describe the CHOOSEPIVOT algorithm for finding a pivot. Explain why choosing the first element of the array can be a very poor choice. (5 marks)

1	<hr/> <hr/> <hr/>
2	<hr/> <hr/> <hr/>

- (c) How does the PARTITION algorithm work? (5 marks)

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>
---

**TURN OVER**

(d) Explain why quicksort uses INSERTIONSORT?

(3 marks)

---

---

---

---

(e) Assuming that PARTITION takes  $n$  operations and that the pivot exactly splits the array in half at each step. Write a recursion relations for the number of partitioning operations,  $T(n)$ . (5 marks)

$T(n) =$

(f) Show that  $T(n) = n \log_2(n)$  satisfies the recursion relation in part (e). (5 marks)

---

---

---

---

---

---

End of question 2

Q2: (a)  $\frac{1}{10}$  (b)  $\frac{1}{5}$  (c)  $\frac{1}{5}$  (d)  $\frac{1}{3}$  (e)  $\frac{1}{5}$  (f)  $\frac{1}{5}$  Total  $\frac{1}{33}$

• Do not write in this space •

**Question 3**

- (a) Show how the numbers 6, 78, 20, 34, 36, 13, 37, 4, 87 would be stored in a simple binary search tree (assuming they were entered in the order given).  
(7 marks)

- (b) What is the typical time complexity of search in a simple binary search tree? What is the worst case time complexity and how can binary search trees be modified to improve the worst case?  
(3 marks)

1	_____
	_____
2	_____
	_____
3	_____
	_____

**TURN OVER**

- (c) Show how the numbers 6, 78, 20, 34, 36, 13, 37, 4, 87 would be hashed using a hash function  $d_2 + 3d_1$  where  $d_1$  is the first (least significant) digit and  $d_2$  the second digit. Show how these would be stored in a hash table using separate chaining. (10 marks)

$d_2d_1$	6	78	20	34	36	13	37	4	87
$d_2 + 3d_1$									
$(d_2 + 3d_1) \% 10$									

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

- (d) Show how the numbers 6, 78, 20, 34, 36, 13, 37, 4, 87 would be stored in a hash table using linear probing assuming the same hash codes. (7 marks)

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	



- (e) What is the disadvantage of linear probing and how can open addressing be modified to overcome this disadvantage? *(4 marks)*

1	<hr/> <hr/> <hr/>
2	<hr/> <hr/> <hr/>

- (f) What type of hashing does Java use and why? *(2 marks)*

<hr/> <hr/> <hr/>
-------------------

End of question 3

Q3: (a) $\frac{7}{7}$ (b) $\frac{3}{3}$ (c) $\frac{10}{10}$ (d) $\frac{7}{7}$ (e) $\frac{4}{4}$ (f) $\frac{2}{2}$ Total $\frac{33}{33}$
---

- Do not write in this space •

**TURN OVER**

**Question 4** This question covers many different topics.

- (a) What is the time complexity of solving the Travelling Salesperson Problem by exhaustive enumeration? *(2 marks)*

---

- (b) Describe the stack abstract data type. *(3 marks)*

---

---

---

---

---

---

---

---

---

- (c) Describe the three methods that make up the iterator interface. *(3 marks)*

1 

---

---

2 

---

---

3 

---

---

---

- (d) Write code for computing the factorial function (1) with recursion and (2) without recursion (do not worry about checking preconditions). (4 marks)

(1)

```
int factorial(int n) { /* recursive */
```

```
}
```

(2)

```
int factorial(int n) { /* non-recursive */
```

```
}
```

- (e) Which is faster and why?

(2 marks)


**TURN OVER**

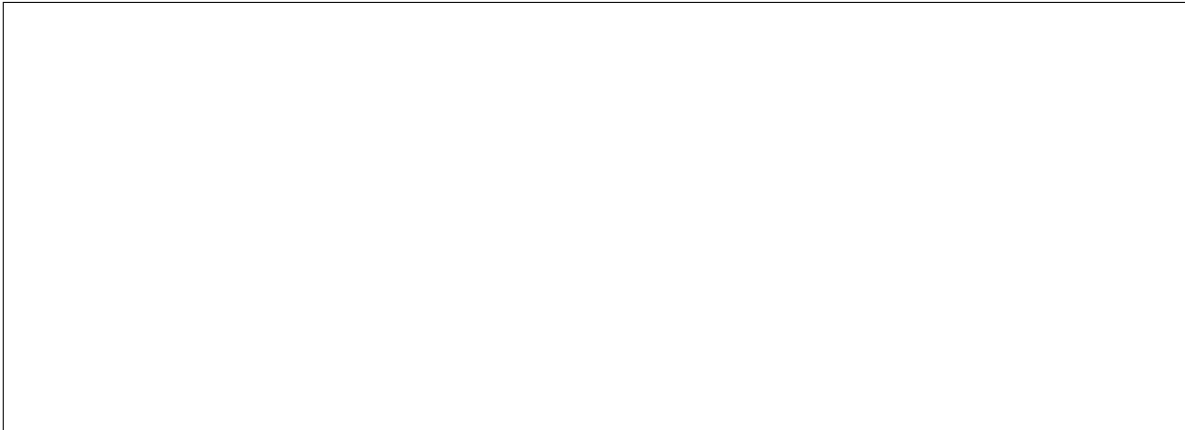
(f) Show how the **heap** shown below is represented as a binary tree.

0	1	2	5	7	4	7	15	6	17
---	---	---	---	---	---	---	----	---	----

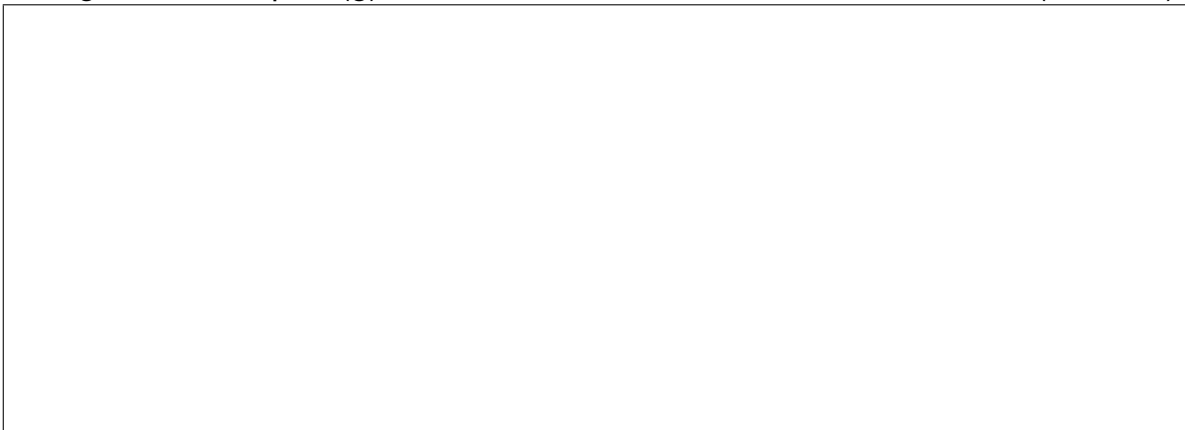
(3 marks)



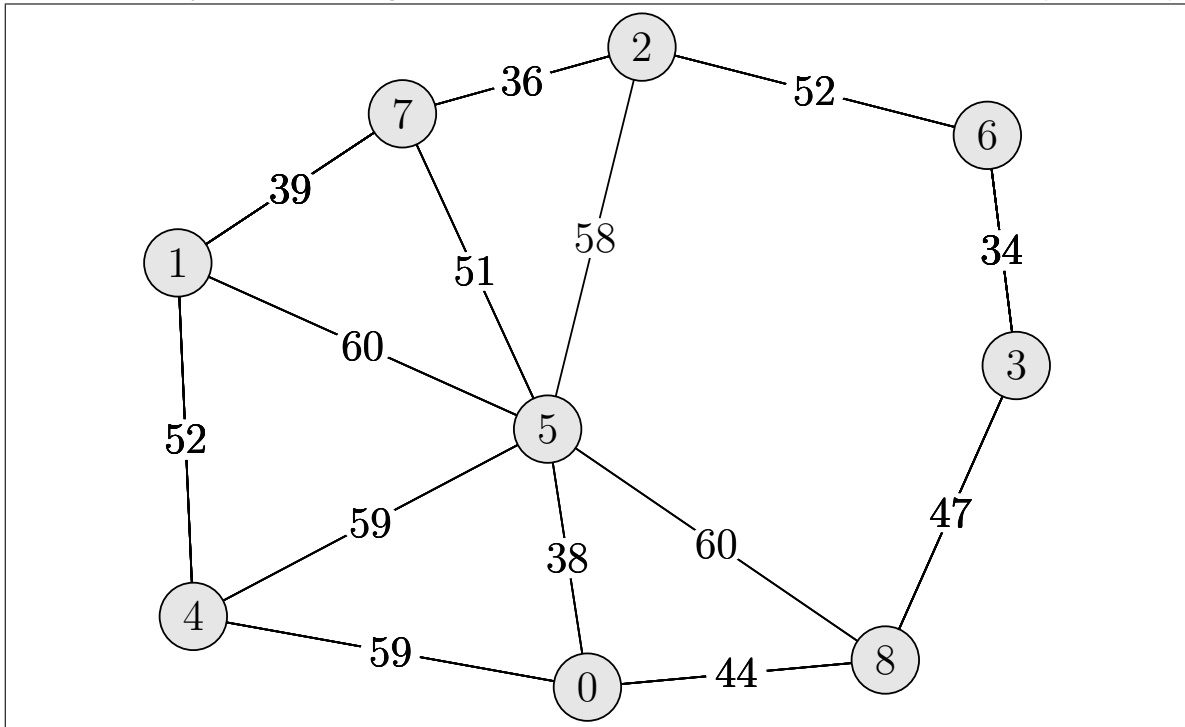
(g) Show what happens if you add 6 to the heap shown in part (f). (3 marks)



(h) Show what happens if you remove the minimum element from the heap generated in part (g). (3 marks)



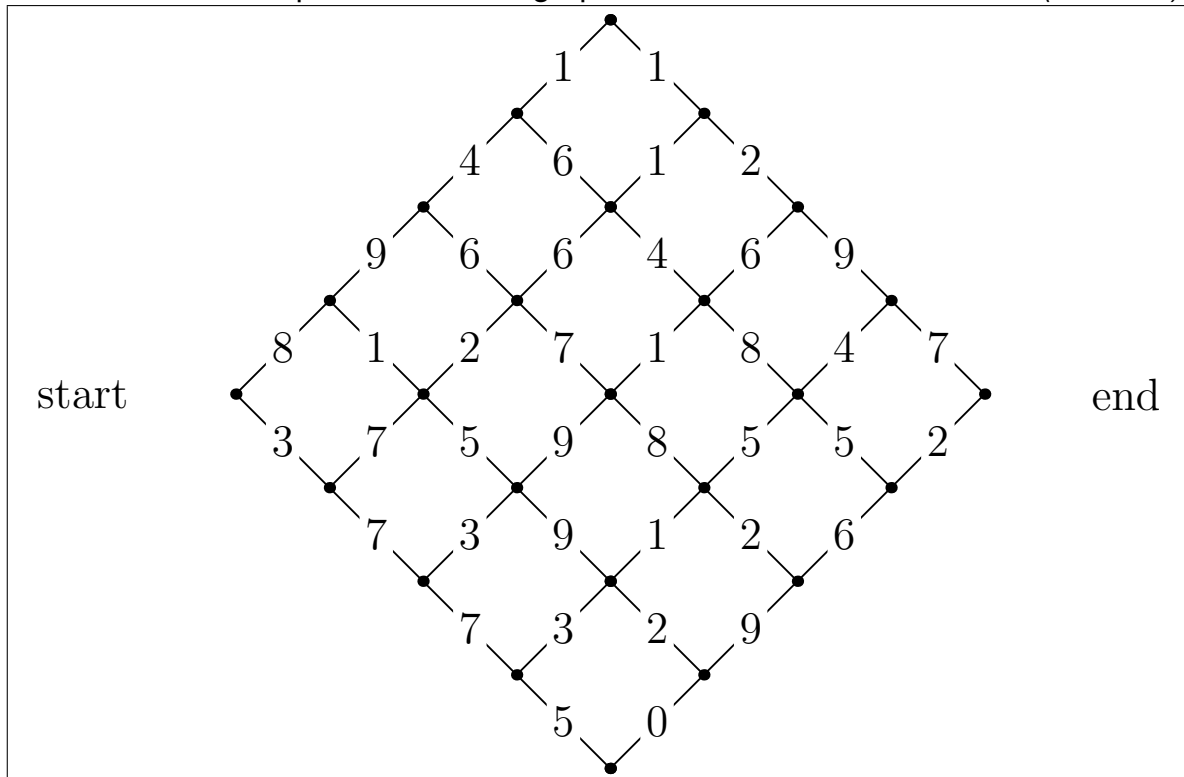
- (i) Show the minimum spanning tree and write down the order of the edges found by Kruskal's algorithm. (4 marks)



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

**TURN OVER**

- (j) Use the dynamic programming forward algorithm to compute the minimum cost of each paths from the left most node to each other node where the cost of moving along an edge is equal to the number shown. An edge can only be traversed from left to right. Use the backwards algorithm to find the minimum cost path across the graph. (6 marks)



End of question 4

Q4: (a)  $\frac{2}{2}$  (b)  $\frac{3}{3}$  (c)  $\frac{3}{3}$  (d)  $\frac{4}{4}$  (e)  $\frac{2}{2}$  (f)  $\frac{3}{3}$  (g)  $\frac{3}{3}$  (h)  $\frac{3}{3}$  (i)  $\frac{4}{4}$  (j)  $\frac{6}{6}$  Total  $\frac{33}{33}$

**END OF PAPER**