UNIVERSITY OF SOUTHAMPTON COMP1009W1

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:

ISS ID:

Question	Marks
1	
2	
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.

 Number of pages 14

Question 1

(a) Fill in the implementation for a simple resizable array. *(10 marks)*

(Test knowledge of variable size arrays, generics and coding.)

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

    public MyArrayList() {
        elements = (T[]) new Object[10];
        noElements = 0;
    }

    public T get(int index) throws Exception {
        if (index>=noElements)
            throw new Exception();
        return elements[index];
    }

    public void addToEnd(T newElement) {
        if (noElements >= elements.length) {
            T[] newArray = (T[]) new Object[2*noElements];
            for (int i=0; i<noElements; ++i)
                newArray[i] = elements[i];
            elements = newArray;
        }
        elements[noElements] = newElement;
        ++noElements;
    }

    public int size() {
        return noElements;
    }
};
```

(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)*

Simple calculation to test understanding of cost of ArrayLists.

8 + 16 + 32 + 64 = 120 copies

(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)*

Given in the notes, but requires understanding.

$$n\,2^m < N \leq n\,2^{m+1} \qquad \textbf{or} \qquad m = \left\lfloor \log_2 \left(\frac{N}{n} \right) \right\rfloor$$

(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)*

$$\begin{aligned} n + 2n + \cdots + 2^m\, n &= n(2^{m+1} - 1) \\ &\leq 2n2^{\log_2(N/n)} - n = 2\,N - n \end{aligned}$$

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

(Test general knowledge about ArrayLists and linked lists)

	ArrayList	**Doubly linked list**
Adding an element to end of list	$\Theta(1)$	$\Theta(1)$
Adding an element to beginning of list	$\Theta(n)$	$\Theta(1)$
Accessing the i^{th} element	$\Theta(1)$	$\Theta(n)$
Searching for and element	$\Theta(n)$	$\Theta(n)$

End of question 1

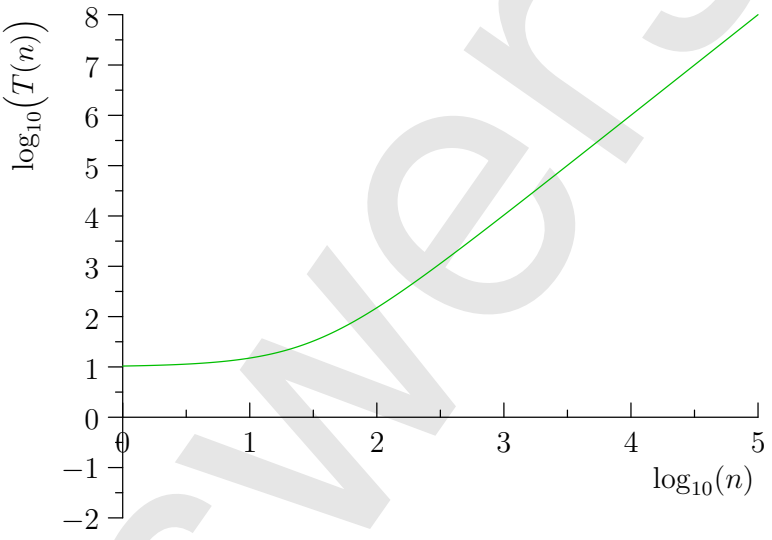
Q1: (a) $\frac{}{10}$ (b) $\frac{}{3}$ (c) $\frac{}{4}$ (d) $\frac{}{8}$ (e) $\frac{}{8}$ Total $\frac{}{33}$

Question 2

TURN OVER

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

$$T(n) = a\,n^c + O(n)$$



From the graph estimate a and c and compute the time it would take to sort 1 000 000 elements. For full marks explain your answers. *(10 marks)*

(Question to test understanding of empirical measure of time complexity. Students should have done this in lab sessions.)

We are only interested in the large n behaviour were we can ignore the $O(n)$ part. Taking logarithms we find $\log(T(n)) = c\log(n) + \log(a)$. Thus the gradient (for large n) is c and the intercept $\log(a)$. The gradient is $c = 2$ and the intercept is $\log(a) = -2$ or $a = 10^{-2}$ so $T(n) = 10^{-2}n^2 + O(n)$. For $n = 10^6$ the time taken will be $T(10^6) = 10^{10}$.

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)*

(Test understanding of qsort.)

(i) A reasonable algorithm is to choose the median of the first, middle and last element of the array.

(ii) Choosing the first element would be poor if the array was already sorted since the split would be completely unbalanced.

(c) How does the Partition algorithm work? *(5 marks)*

The algorithm works by starting from either side of the array. It finds the first element on the left side that is greater than the pivot and finds the first element on right side that is less than or equal to the pivot. The two elements are swapped. This is repeated until the two searches meet.

(d) Explain why quicksort uses InsertionSort? *(3 marks)*

For small array sizes insertion sort is faster than quicksort.

(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 patitioning operations, $T(n)$. *(5 marks)*

(Test knowledge of computing time complexity.)

$$T(n) = 2T(n/2) + n$$

TURN OVER

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

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &= 2\left(\frac{n}{2}\log_2\left(\frac{n}{2}\right)\right) + n \\ &= n\log_2\left(\frac{n}{2}\right) + n \\ &= n\left(\log_2(n) - \log_2(2)\right) + n = n\log_2(n) \end{aligned}$$

End of question 2

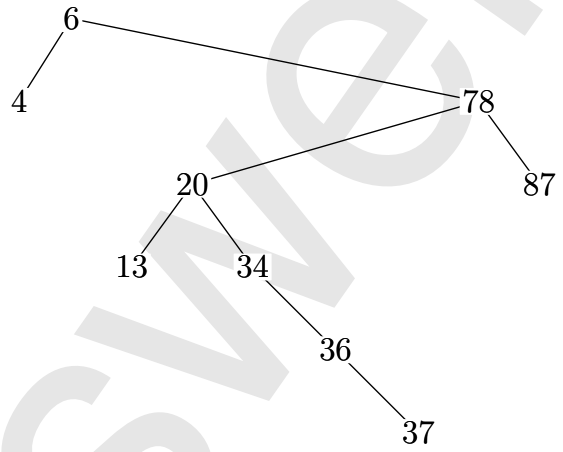
Q2: (a) ___/10	(b) ___/5	(c) ___/5	(d) ___/3	(e) ___/5	(f) ___/5	Total ___/33

• 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)*

(Test understanding of sets.)



(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)*

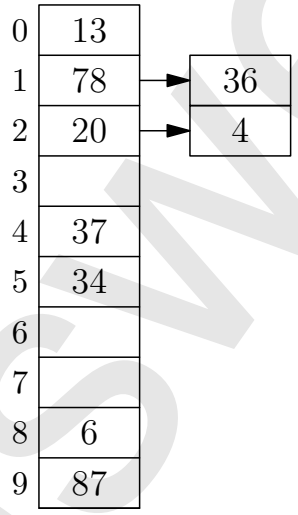
(i) $O(\log(n))$

(ii) n **when the numbers are entered in order**

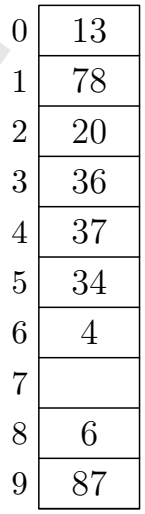
(iii) The tree can be balanced to make sure the deepest branch is $O(\log(n))$

(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
$d_2 + 3d_1$	18	31	2	15	21	10	24	12	29
$(d_2 + 3d_1)\%10$	8	1	2	5	1	0	4	2	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)*



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

(i) Linear probing is susceptible to develop primary clusters where large contiguous regions of the hash table are used.

(ii) Either quadratic probing or double hashing is used so if there is no space a larger jump is made.

(I would also accept a discussion of the problem of deleting elements.)

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

Java uses separate chaining as it is more robust at high loading factors.

End of question 3

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

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)*

(Question to test all aspects of the course)

$O(n!)$

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

It is a model of a first in last out memory with the methods `push(item)` which pushes `item` onto the stack, `peek()` which returns the item on the top of the stack, `pop()` which returns the item at the top of the stack and removes it, and `isEmpty()` which returns true if the stack is empty and false otherwise.

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

(i) **`hasNext()` returns true if there is another element in the container to iterator over.**

(ii) **`next()` returns the next element and throws an exception if there is no next element.**

(iii) **`remove()` removes the last element called by `next()` and throws and exception if `next()` has not been called.**

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

(Answer either in Java or pseudo code acceptable.

```
int factorial(int n) { /* recursive */
  if (n==0)
      return 1;
  return n*factorial(n-1)
}


int factorial(int n) { /* non-recursive */
  int ans = 1;
  for(int i=2; i<=n; i++) {
      ans *= i;
  }
}
```

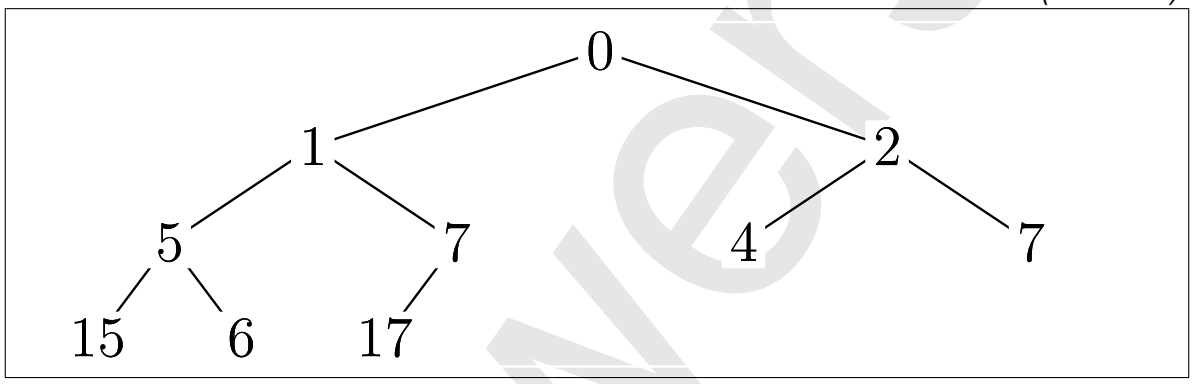
(e) Which is faster and why? *(2 marks)*

The non-recursive version is fast as it does not involve recursive function calls.

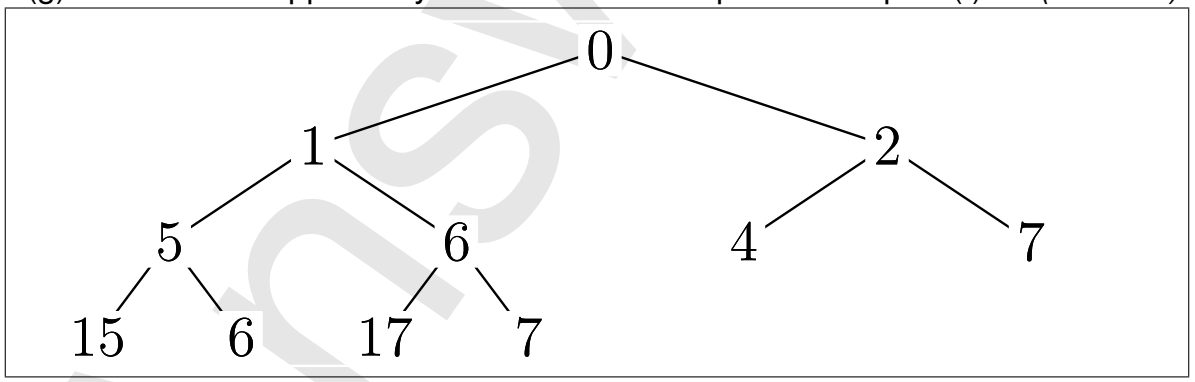
(f) Show how the **heap** shown below is represented as a binary tree.

0	1	2	5	7	4	7	15	6	17

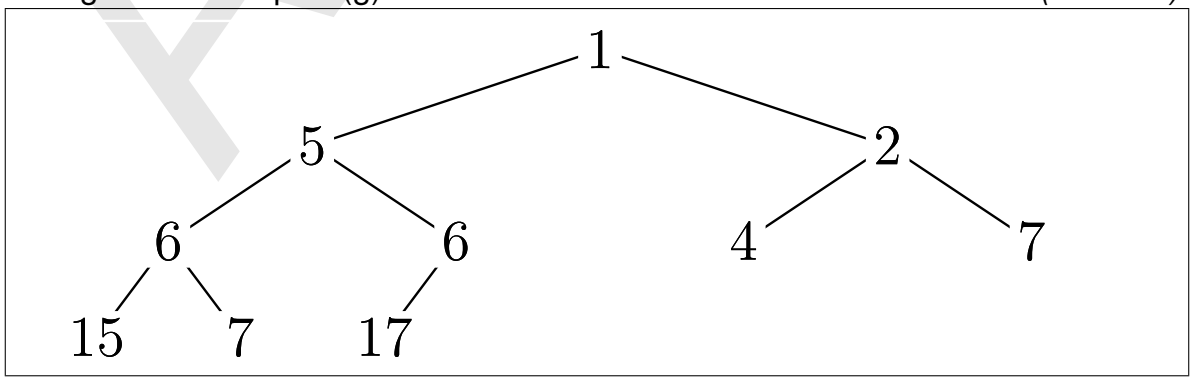
(3 marks)



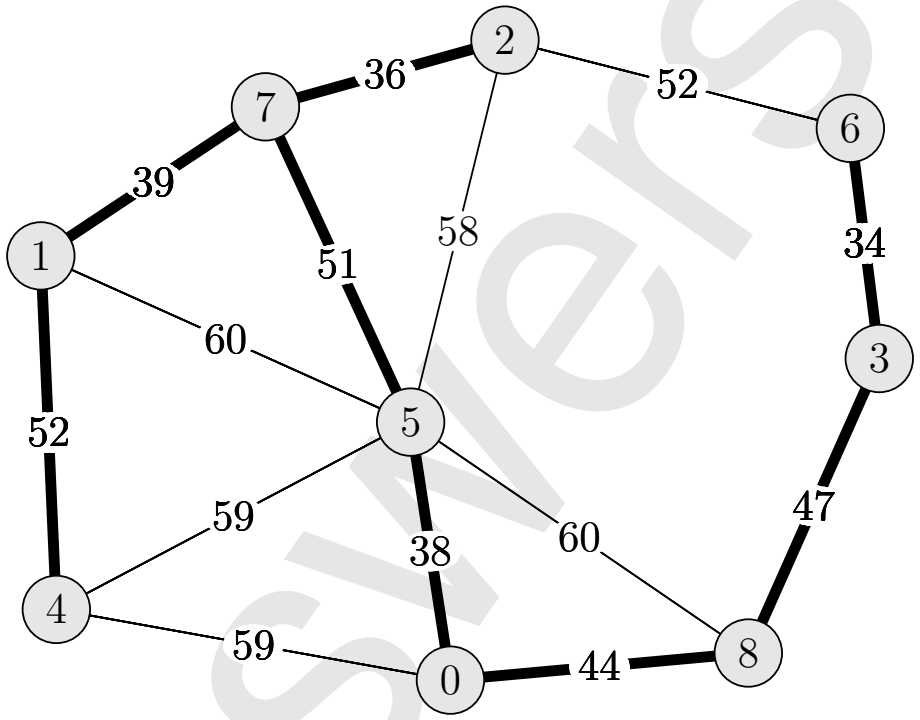
(g) Show what happens if you add 6 to the heap shown in part (f). *(3 marks)*



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

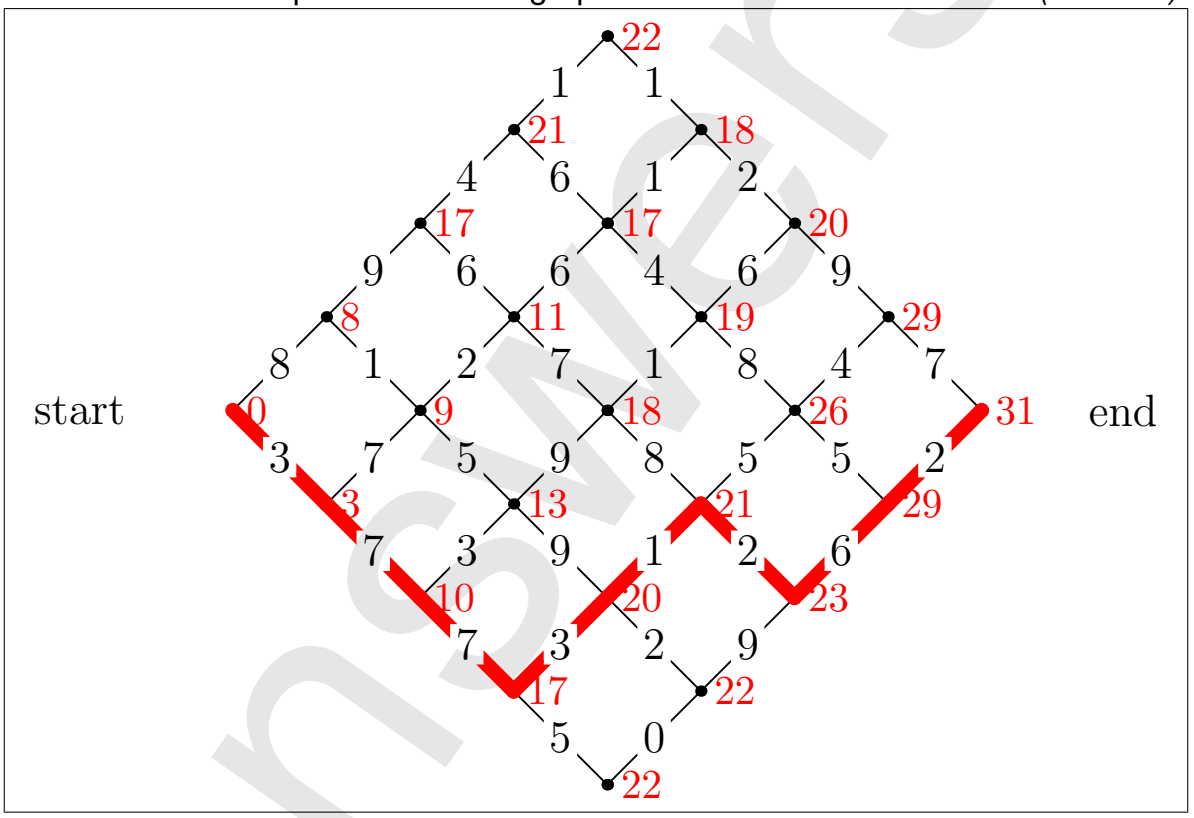


(i) Show the minimum spanning tree and write down the order of the edges found by Kruskal's algorithm. *(4 marks)*



1.	$(3, 6)$	2.	$(2, 7)$
3.	$(0, 5)$	4.	$(1, 7)$
5.	$(0, 8)$	6.	$(3, 8)$
7.	$(5, 7)$	8.	$(1, 4)$

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

END OF PAPER