1. **Multiple Choice** *(10 marks)*
Each question in this section is worth 2 points, it has exactly one correct answer and you are not penalized for incorrect answers. Write clearly. If we cannot understand your choice, you will receive no credit.

(a) *(2 marks)*
The worst case time to delete a single element from a singly-linked list with n nodes is:
(i) $O(1)$ (ii) $O(n)$ (iii) $O(n \log n)$ (iv) $O(n^2)$

A *complete* binary tree is a binary tree in which all leaves are at the same depth.

(b) *(2 marks)*
The best case time to delete a single element from a complete binary search tree with n nodes is:
(i) $O(1)$ (ii) $O(\log n)$ (iii) $O(n)$ (iv) $O(n \log n)$

(c) *(2 marks)*
The worst case time to insert a single element in a complete binary search tree with n nodes is:
(i) $O(\log n)$ (ii) $O(n)$ (iii) $O(n \log n)$ (iv) $O(n^2)$

(d) *(2 marks)*
The number of nodes in a complete binary tree with eight levels is:
(i) 127 (ii) 128 (iii) 255 (iv) 256

(e) *(2 marks)*
What is the maximum number of levels (root is in level 1) of an AVL tree with 7 nodes.
(i) 3 (ii) 4 (iii) 5 (iv) 6

2. **Complexity** *(20 marks)*
Give a tightest possible bound for the Big-Oh runtime complexity of these two functions in terms of the value of the input parameter n.
Results without derivations do not receive any points.

(a) *(5 marks)*

```
1   int easy(int n, int *a) {
2       int i;
3       for (i = n; i > 1; i = i/2) {
4           a[i-1]++;
5       }
6   }
```

(b) *(5 marks)*
(Hint: The complexity of `simple(n)` *is* $T(n)$*.)*

```
1   int simple(int n) {
2       int a, b, c;
3       if (n <= 1)
4           return 0;
5       else {
6           a = simple(n-1);
7           b = simple(n-1) / 2;
8           c = simple(n % 2);
9           return a + b + c;
10       }
11   }
```

(c) *(10 marks)*

The following table lists a set of data structures in its rows. Each of these data structures can have a new element inserted into it. If one assumes that the most efficient algorithm possible is used, list the complexity of insertion for each data structure, using big-Oh notation, for the best and worst case situations.
Assume n elements have already been inserted before.

	best case	worst case
unsorted array		
sorted array		
unsorted single-linked list		
sorted single-linked list		
binary search tree		

3. **Lists and Trees** *(20 marks)*

Mr. X tries to implements a binary search tree (BST) using pointers. Each node only contains a key and two pointers leading to its children. As he did not take COMP104 and not COMP171 either, after deletion of a leaf the pointer originally pointing to that leaf points to some other node in the tree now instead of pointing to NULL.

The BST has n nodes, whereby the value of n is unknown.

(a) *(5 marks)*

Design an $O(n)$ algorithm to determine if the tree contains a cycle. You may use $O(n)$ extra space.

Give modest comments.

(b) *(5 marks)*

Repeat part (a), but use only $O(1)$ extra space.
(Hint: Use two iterators that are initially at the root of the tree but traverse the tree at different speeds.)
Give modest comments.

(c) *(10 marks)*

Assume now that the BST is balanced, so that its depth is at most $\log n$. The faulty pointer that is pointing to some other node in the tree instead of pointing to NULL does not necessarily close a cycle. Design an algorithm using only $O(\log n)$ extra space to discover this incorrect pointer.

Give modest comments.

4. **Stacks and Queues** *(20 marks)*

(a) *(10 marks)*

Provide a recursive pseudo-code routine that reverses the elements in a queue, **Q**, using only the operations `isEmpty`, `size`, `delete`, and `add`.

The operation "`add`" appends an element to the rear end of the queue.

The operation "`delete`" removes an element from the front of the queue.

Note that recursion gives you an implicit stack; you may not use any explicit data structures.

Your program must run in $O(n)$ time.

Give modest comments.

(Hint: The solution has less than 10 lines.)

Example:

Input Queue Q:		Output Queue Q:
$5 \to 7 \to 8 \to 9 \to 3$	$\Longrightarrow$	$3 \to 9 \to 8 \to 7 \to 5$

(b) *(10 marks)*

Provide a **non**-recursive pseudo-code routine that reverses the elements in a queue, **Q**, using only the operations `isEmpty`, `size`, `delete`, and `add` (as defined in part (a) above) and a second queue **R**.

You may not use any other data structures, just **Q** and **R**.

You can assume there is no memory limit for **Q** and **R**.

Your program must run in $O(n^2)$ time.

Give modest comments.