# Sample Paper

- If using a diagram to show e.g. the structure of a doubly-linked list, you need to also show the code underneath a little, by showing the instance variables.
- The defining characteristic of the Queue ADT would be that it's FIFO.
- For pseudocode answers, you just need to convince him that you know how things work. Remember with removing nodes from linked structures that the order of operations is important.
- Parts that ask for unseen material may ask you for the most efficient solution, in which case you'll get marks for more efficient solutions, or they may just ask you to give the worst case complexity for your solution, in which case you won't lose marks for your solution being inefficient.
- You can define the height of a node using the recursive definition or as ~ "the longest path to a leaf, measured by the number of edges".
- For defining an AVL tree – "An AVL tree is a binary search tree where no node is unbalanced" – you'll need to define an unbalanced node.
- When asked for pseudocode for rotation, you should draw a diagram of how the tree is changed, as you'll get partial credit for that even if your pseudocode is incorrect.
- When writing code for rebalancing AVL trees, you can write just for the straight case and zigzag case in one direction, and add a comment saying "and the 2 symmetric cases".
- There will be questions on the exam that require manipulating trees, whether that's binary search trees or heaps.
- For removing the root node from an AVL tree, find the highest left descendant and move it into the position, then delete it's old node and

perform any rebalancing needed.

- Example description of a HashTable:
    - keys must be immutable type
    - hash converts key to an int
    - compression converts in to an index in the array
    - if two (k, v) items go in the same cell, we have a container (queue, stack, etc.) [this for a bucket example]
    - accessing involves searching that container
    - resizing array-based list when `n/N > threshold`
        - n is the number of items in the map
        - N is the size of the list
    - resizing involves rehashing everything
- For a 10-mark question, you will need a fair bit of detail.
- We use a hash table for fast access time. Expected size below a threshold, so expected complexity is O(1).

---

# My Solution

## Question 2

### (viii) [listed on paper as the second part (vii)]

- The complexity of searching an AVL tree is O(logn), as we've guaranteed the tree is balanced, and the furthest we'll have to search is down the longest path.
- The complexity of adding to an AVL tree is O(logn), as we have to first check if the element is there, then add the new element, then rebalance back up the tree. Worst case will be about `2logn` steps, which is O(logn).