Question 1. Grader: Aniket

**Rubrik:** +1 for correct answer, +3 for correct explanation without any example or correct explanation with some small mistake or an example without proper explanation, +1 for correct example in addition to the explanation.

**Common Mistake:**

Using Backtracking:Showing 2f(n) = O(2g(n)) implies f(n) = O(g(n)) and drawing conclusion based on that.  
Writing xa\*b = xa \*xb

Question 2. Grader: Shravan

**Rubrik:** +1.25 for each correct blank, +1.25 for correct time complexity.

**Minor Inaccuracies:** Many students have incorrectly used class functions described above. There has been no penalty for incorrect use this time. As an example, simply writing isLast(q) would be incorrect because isLast() is a function of the list class and hence can be accessed only by objects of this class. So, a correct function call would be L.isLast(q)

Question 3. Grader: Manav

**Rubrik:** +5 for fully correct answer, -1 for incorrect complexity, -4 for inefficient solution (as major question is efficient solution), +0.5 is for first blank, +1.5 is for second blank, +2 is for third blank.

**Comments:** Remarks as follows:  
**- for the first blank:** It is considered correct if it acts as base case for every possibility.  
**- for the second blank:** It must traverse into one branch only.  
**- for the third blank:** One must use size() function to count nodes of one branch and traverse only in one branch.  
**- for complexity:** Marks will only be given if your solution is correct.  
**- for inefficient solution:** 1 mark will be given for inefficient solution only if it is fully correct.

Question 4.1. Grader: Sourav

**Rubrik:**+3 marks for correct expression, +2 marks for correct explanation, +1 marks for partially correct explanation, +1 for partially correct ideas

**Correct Answer:** (2^d – 1) + 2^(d-1)  
Let d be the depth of the shallowest leaf. Upto depth d-1, the tree should be complete. Each of the nodes in level d-1 should have only 1 child (to minimize the number of nodes). Also, if some node has 0 children, the shallowest leaf would not be at depth d. In case of recursive answer, you should write the correct expression and the correct base case.

**What is a correct explanation?**  
When you have completely explained the idea that upto level d-1, it should be a complete tree and there should be one child for each of the nodes in level d-1. In case of recursive expression solution, you should have explained how you got the recursive expression.

**Comments**

* If your expression is correct but there’s a mistake in final calculation (like the final GP Sum or solving of recursive expression), we have given the marks.
* If the depth of shallowest leaf is not d (eg the solution where the student supposes complete tree upto depth d-1 and just one node at depth d has been given 0 marks)
* No mark is given if you have written just the expression of number of nodes in compete binary tree without giving any explanation or mentioning that it is a complete binary tree.
* Partially correct ideas includes when the student concludes
  + that all the leaves should be at the same depth which the shallowest leaf depth. This also includes the complete binary tree ***upto depth d.***

Question 4.2. Grader: Harshil

**Rubrik:** +2 for correct answer, +3 for correct explanation.

**Correct Answer:** 2n (n is last 5 digits of your entry number). We have also awarded full points for 2n + 1 and 2n – 1 (or any answer off by a constant).

**Common Incorrect Answers:** n, n + 1, n – 1, infinity. Zero marks are awarded in this case.

**Common Incorrect Explanation:** Proof by example or proof by diagram does not work unless you have generalised the example. Showing a particular tree of small height does not prove it for a general n.

Request a regrade **if:**  
1. You answer is within a constant additive error of correct answer and are not awarded 2 points.  
2. You have provided a generalised explanation and not been awarded 3 points.

Question 5. Graders: Aman, Anshuman, Harshavardhan

1) Before accessing children of any node (say at position “pos”) in the heap it is necessary to first check if the children exist or not using conditions like 2\*pos+1 < self.levelorder.size () or 2\*pos+2 < self.levelorder.size (). Not doing so can lead to Index Error. 2 marks were deducted if this mistake is made.

2) Modifying the original heap is marked incorrect as you were supposed to implement an accessor method.

3) Approaches adding all elements of levelorder from 0 to k/k+1/k+2 in the aux heap are incorrect because the k^th minimum element can exist beyond these indices in the levelorder array.

4) Adding all elements of levels up to the level k of the original into the aux heap is very computationally expensive (j^th iteration is performing 2^j enqueue operations). No marks are awarded for this implementation as you were expected to complete the operation in O(k\*logk) time.

5) levelorder[k] does not work; elements in `levelorder` array are not sorted. Level order means the array stores elements level by level starting from left of each level. For a heap, the level order array will not necessarily have its elements sorted.