Q1 a. Output is as follows:

0 (0.5 marks)

Child. 10 (0.5 marks)

Parent (0.5 marks)

0.5 marks for the correct order.

Q1 b.

Output pattern: 0 1 2… 2… 1 2… 2… Because the question mentions What's printed in what order (1 mark)

Mentioning Pre-order traversal, each node prints its depth or if n is the number of nodes at a depth d then d is printed n times. (3 marks)

Q3

'Suppose the pivot is the first element' is a flaw since it does not guarantee worst case partitioning. (2 marks)

'Partition takes = kn+c' is a flaw since there is no support to it in the proof because it must be <= kn+c (2marks)

Q4

Mentioning that 'An algorithm must search through all k+1 elements' is a flaw. (2 marks)

Reasoning for above flaw is that to find the element which is going to be obtained in the (k+1)th comparison, we need not search through all the k+1 elements again because binary search algorithm divides the search domain by half in every iteration. So we don't need to search through all k+1 elements. (2 marks)

Q9

-intution

1) pick the middle element.

2) If arr[mid+1]>arr[mid] and arr[mid-1]<arr[mid] , we recur for right

half.

3) If arr[mid+1]<arr[mid] and arr[mid-1]>arr[mid] , we recur for left

half.

4) If arr[mid+1]<arr[mid] and arr[mid-1]<arr[mid] , return mid

-algo

find_k(arr[],start,end)

{ mid=(start+end)/2

```
 if(arr[mid+1]>arr[mid] && arr[mid-1]<arr[mid])
       find_k(arr[],mid+1,end)
   else if(arr[mid+1]<arr[mid] && arr[mid-1]>arr[mid] )
        find_k(arr[],start,mid-1)
   else if(arr[mid+1]<arr[mid] and arr[mid-1]<arr[mid] )
         return mid
   else
     return -1  //no such element exist

}
```

-analysis

At each iteration we reduce the size of an array to half, and at each iteration constant time operation is being done(comparison operation), so clearly ,

tc=O(log(n))