I had two rounds of interview after the internship.(Problem solving + Data structures and Algorithms)
They considered interview performance + the job done during internship.
Both the rounds were Face to face interviews.

Round 1 (Problem Solving):

Find all such numbers such that they satisfy the following property.
1) Seven digit number.
2) All digits must be distinct.
3) Product of first three,middle three and the last three must be equal.
Ex : a b c d e f g
a*b*c = c*d*e = e*f*g

Solution:

We can modify the relation as
a*b = d*e
and
c*d = f*g

So this has reduced the problem now. We have to find numbers such that it can be factorized into two different ways and these two digits must be within 0 to 9.

There are only five numbers which satisfy the above property
1*6 = 2*3 = 6
1*8 = 2*4 = 8
2*6 = 3*4 = 12
2*9 = 3*6 = 18
3*8 = 4*6 = 24

So one possible solution to the problem is
1892436, with 1*8*9 = 9*2*4 = 4*3*6 = 72

However I didn’t give him the complete solution.

Round 2(Algorithms and Data Structures):

1) Design a data structure for doing the following operations in constant time.

- Insert(insert a given element).
- Delete(delete a given element).
- Search(given an element, say if it is there already).
- Find any(Doesn’t have any parameter,just return any element that you have inserted already).

Solution:
I gave him a solution of having a Doubly Link List and a HashMap for doing this.He was satisfied.

i) Insert:
Insert into hashmap and link list.HashMap will store the address of the node in the linklist.
ii)Delete: Get the exact node from hashMap and delete it.
iii)Search: simple hashmap retrieval.
iv)Find any: return head of the link list.

2) Find the second maximum in less number of comparisons.

Solution:

Tournament method.

Make a small change to the tournament method that we have already learnt in lab. That will yield us second maximum in N+logN comparisons.

In detail:

Divide the array into parts like merge sort and construct a tree which will store the max at each node. This you can do in N comparisons.
Then usually you will traverse down the tree comparing all the elements which we have compared with the max element to find the second maximum. For doing this easily, have the small element on one side of the tree while you are going up(finding the max). So when you come down, you can go in a particular path without comparing where the less element is. So this saves logN comparisons. Then traverse down the tree to compare all elements which we have compared already with max.This is traversing the tree from root to leaf so LogN comparisons. So N + Log N comparisons. :)